VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

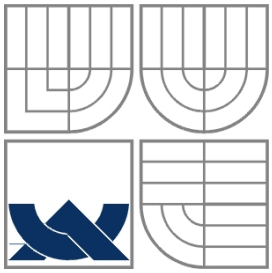
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMUNITNÝ HUDOBNÝ PORTÁL

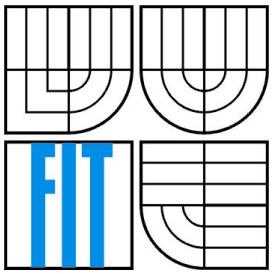
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ANDREJ NOVOSAD
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMUNITNÍ HUDEBNÍ PORTÁL

COMMUNITY MUSIC PORTAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ANDREJ NOVOSAD
AUTHOR

VEDOUCÍ PRÁCE — ING. OTA JIRÁK
SUPERVISOR

BRNO 2010

Abstrakt

Práce se zabývá analýzou, návrhem a implementací webového portálu, který je zaměřený na hudební akce elektronických tanečních stylů a na způsob, kterým se tyto akce filtrují pro uživatele.

Abstract

Thesis discusses the analysis, design and implementation of a web portal, which is focused on music events of electronic dance styles and the way they are filtered for the user.

Klíčová slova

Webový portál, hudba, PHP, Zend Framework, JavaScript, jQuery, MySQL, AJAX, JSON, HTML, XHTML, CSS, filtrování obsahu

Keywords

Web portal, music, PHP, Zend Framework, JavaScript, jQuery, MySQL, AJAX, JSON, HTML, XHTML, CSS, content filtering

Citace

Novosad Andrej: Komunitný hudobný portál, bakalářská práce, Brno, FIT VUT v Brně, 2010

Komunitný hudobný portál

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Oty Jiráka.
Další informace mi poskytli Ivan Novosad a Marián Hacaj.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Andrej Novosad
18.05.2010

Poďakovanie

Chcel by som sa poďakovať svojmu vedúcemu Ing. Otovi Jirákovi za svoj čas, ktorý mi bol ochotný venovať aj mimo pracovný týždeň a za rady, ktoré mi poskytol pri analýze projektu a pri vytváraní tejto dokumentácie.

Obsah

1 Úvod

Pre moju bakalársku prácu som si zvolil tému, vďaka ktorej by som sa mohol ďalej rozvíjať v oblasti informatiky, ktorá ma najviac baví a zaujíma, a v ktorej mám najväčšiu motiváciu dosahovať čo najlepších výsledkov. Zároveň sa vybraná téma spája s mojim osobným životom a záľubami, čo som mohol uplatniť a využiť v prospech tohto projektu pri jeho navrhovaní a vývoji.

Zvolil som si za úlohu navrhnúť a implementovať webový portál, ktorý bude zameraný na štýly elektronickej tanečnej hudby a ich komunity. Mal by ponúkať zoznam hudobných akcií (prípadne iných záležitostí) najznámejších elektronických štýlov. Tieto akcie bude stránka schopná užívateľovi odporúčať podľa toho, čo užívateľ na stránkach bude robiť. Vďaka tomu sa aplikácia o užívateľovi bude stále presnejšie dozvedať, aké konkrétne štýly ho bavia, čo má vlastne rád. Užívatelia si teda takýmto spôsobom vytvoria spoločnú databázu interpretov, klubov a hudobných štýlov a akcií a s tým súvisiace štatistiky, vďaka ktorým sa budú dať jednotlivý interpreti a kluby hodnotiť.

Celá webová aplikácia by mala byť odetá v peknom modernom dizajne s príjemným a praktickým užívateľským rozhraním za použitia moderných technológií a trendov. Kladený by mal byť dôraz na rýchlosť stránok, intuitívnosť ovládania a na bezpečnosť užívateľov aj celej aplikácie.

2 Použité technológie

V tejto kapitole sú spomenuté technológie, ktoré som použil na tvorbu tohto projektu. Okrem základnej charakteristiky je pri každej technológií načrtnuté, z akého dôvodu a akým spôsobom som sa ju snažil využiť.

2.1 HTML, XHTML

HTML alebo HyperText Markup Language je značkovací jazyk pre webové stránky. Pomocou neho je možné vytvárať štruktúrované dokumenty s vyznačenou sémantikou pre text. HTML bolo definované pomocou *SGML* (Standard Generalized Markup Language), ktorý predstavuje veľmi flexibilný *framework* pre značkovacie jazyky [1].

XHTML (Extensible HyperText Markup Language) je značkovací jazyk definovaný pomocou *XML*, čo je prísnejšia verzia (podmnožina) SGML. Vzniklo, aby učinilo HTML ľahšie rozšíriteľným jazykom. XHTML verzia 1.1 je odporúčaná konzorciom W3C. Dokumenty v tomto projekte som značkoval podľa špecifikácií XHTML, aby boli *validné*, čo zaisťuje konzistenciu dokumentu [2].

Využil som nový *DOCTYPE*: `<!DOCTYPE html>`. Výhoda tohto DOCTYPE-u je tá, že umožňuje písať dokumenty už teraz v HTML 5 (aj keď to daný prehliadač nemá implementované), čo je dobrý krok z hľadiska budúcnosti [3].

2.2 CSS

Cascading Style Sheets (CSS) [4] je jazyk, ktorým možno definovať *prezentačnú sémantiku* dokumentu napísaného pomocou značkovacieho jazyka, teda jeho vzhľad a formátovanie. Jeho hlavný význam je na oddelenie obsahu dokumentu od spôsobu jeho prezentácie. Toto oddelenie môže zlepšiť dostupnosť obsahu a umožňuje lepšiu flexibilitu a kontrolu v špecifikovaní vzhľadu dokumentu. Pomocou CSS je taktiež možné naštýlovať dokument tak, aby ho prehliadače prezentovali rovnako, alebo aspoň čo možno najbližšie pôvodne zamýšľanému vzhľadu. Preto som všetok vzhľad stránok definoval pomocou CSS [4].

Na odstránenie značného množstva nejednoznačných interpretácií prezentácie stránok rôznymi internetovými prehliadačmi som použil tzv. *reset*. Reset je nastavenie počiatočných štýlov značiek na jednotnú hodnotu ako napríklad okraje, odsadenia, veľkosti fontov atp [3].

Najnovšie verzie prehliadačov podporujú CSS 3, ktoré je práve vo vývoji. V tomto projekte som použil niektoré jeho nové vlastnosti ako zaoblenie rohov, tiene, prípadne iné.

2.2.1 CSS Framework

Existuje niekoľko frameworkov pre CSS, ktoré môžu urýchliť prácu, prípadne efektívnejšie odstrániť nekonzistencie prehliadačov. Spočiatku som mal v pláne použiť *Blueprint framework* [5], ktorý je (ako väčšina ostatných) založený na tzv. „*grid-layout*" prístupe k dokumentu, teda všetky elementy sa umiestňujú do mriežky. Postupne som zistil, že tento prístup nie je veľmi odlišný od zastaraného tabuľkového rozvrhnutia stránky, nie je stavaný na vytvorenie takej šablóny, ktorá by sa prispôsobovala veľkosti okna a navyše mieša prezentáciu so značkovaním.

CSS framework deklaruje pravidlá, ktoré je nutné sa naučit a dodržiavať, a ktoré by som všetky nevyužil. Rozhodol som sa teda CSS framework nepoužiť a všetky štýly si vytvoriť sám.

2.2.2 Browser Sandbox

Na porovnávanie toho, ako rôzne internetové prehliadače interpretujú CSS, som použil online službu *Browser Sandbox* [6], ktorá umožňuje rýchlo a jednoducho pustiť ktorýkoľvek z najrozšírenejších internetových prehliadačov vrátane ich rôznych verzií bez toho, aby museli byť nainštalované. Sú to konkrétne Internet Explorer, Firefox, Chrome, Safari a Opera. V najnovších verziách týchto prehliadačov portál vyzerá konzistentne, podľa pôvodných predstáv. Niekoľko rozdielov a problémov, ktoré som nestihol opraviť, sa dá zaznamenať v prehliadačoch Internet Explorer 6, Opera 9, Firefox 2 a v ich starších verziách.

2.3 JavaScript

JavaScript [7] je multiplatformový objektovo orientovaný skriptovací jazyk. Používa sa najmä vo forme klientského JavaScriptu, ktorý je implementovaný ako súčasť internetového prehliadača. Pomocou neho je možné vytvárať lepšie užívateľské prostredia a dynamické stránky. Bol vytvorený firmou Netscape. Štandardizovaný bol v roku 1997 organizáciou ECMA (Europen Computer Manufacturers Association) a postupne nasledovali ďalšie štandardizácie [7][8].

V tejto práci som JavaScript využíval práve z dôvodu spríjemnenia a rôznych vylepšení užívateľského rozhrania. Využil som na to knižnicu *jQuery* [10] a technológiu *AJAX* [15].

2.4 jQuery

jQuery [9] je rýchla a veľmi účinná knižnica pre JavaScript, ktorá značne urýchľuje a zjednodušuje prácu. Kladie dôraz na interakciu medzi JavaScriptom a HTML. Pomocou tejto knižnice je možné veľmi jednoducho prevádzať väčšinu akcií, ktoré by sme kedy potrebovali ako napr. prechádzať a modifikovať *DOM* štruktúru dokumentu, manipulovať s CSS, spracovávať udalosti, vytvárať animácie a AJAX interakcie [9].

Funkcionalitu jQuery je možné rozšíriť pomocou pluginov. Z nich som využil *autocomplete, sliders* a *timers*.

2.5 PHP

Jadro portálu je napísané v jazyku PHP [10][11]. Je to široko používaný, univerzálne využiteľný skriptovací jazyk, ktorý je špeciálne navrhnutý pre vývoj dynamických webových stránok a aplikácií. Za týmto účelom je vsadený do HTML a interpretovaný webovým serverom pomocou PHP procesorového modulu, ktorý generuje webový dokument, ktorý je následne odoslaný klientovi. Vytvoril ho *Rasmus Lerdorf* v roku 1995, odkedy je stále ďalej vyvíjaný skupinou *The PHP Group* a jeho najaktuálnejšia verzia je 5.3.2 [10][11].

Nakoľko bolo treba vytvoriť pomerne rozsiahli projekt, bolo vhodné si vybrať a použiť nejaký PHP framework. Existuje niekoľko PHP frameworkov, vďaka ktorým je možné začať pracovať na projekte s už pripravenou základnou a najpoužívanejšou funkcionalitou, čo šetrí čas, pomáha tvoriť stabilnejšie aplikácie a redukuje množstvo opakujúceho sa kódu. Zvolil som si tento projekt vytvoriť pomocou *Zend Framework* [12], ktorý je momentálne vo verzií 1.10.4 a potrebuje PHP 5.2.4 alebo novšie.

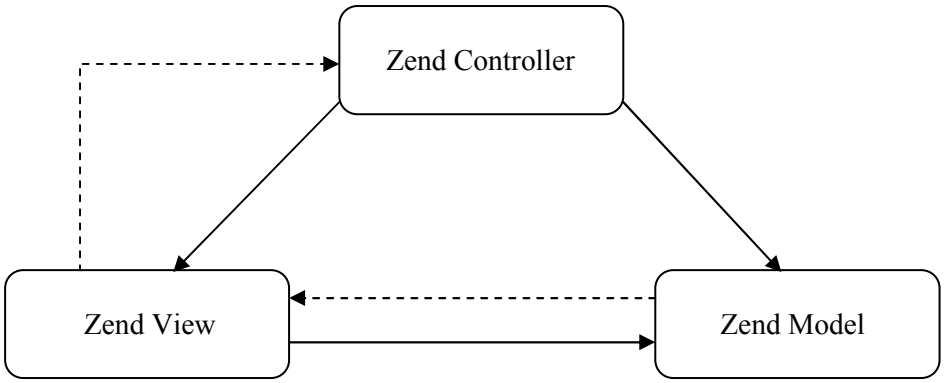
2.6 Zend Framework

Zend Framework je otvorený (*open-source*), objektovo orientovaný framework pre PHP 5. Skladá sa z mnoho komponentov, ktoré je možné takmer nezávisle využívať. Je to jeden z najobľúbenejších PHP frameworkov, má obrovskú podporu komunity, je zameraný na vývoj bezpečných, spoľahlivých a moderných *web 2.0* aplikácií [12]. Má rozsiahlu dokumentáciu, takže k akémukoľvek problému, ktorý sa naskytol, nebolo ťažké nájsť riešenie.

Okrem viacerých výhod a odporúčaní som si tento framework vybral aj preto, lebo už mám s ním aspoň nejaké skúsenosti a chcel som v ňom nadobudnúť ďalšiu prax.

2.6.1 MVC

Základnou ideou za PHP frameworkami je tzv. *MVC model,* ktorý značí ***M**odel*, ***V**iew* a ***C**ontroller*. Je to stavebný vzor v programovaní, ktorý oddeľuje logiku aplikácie (*controller*), prácu s dátami a interakciu s databázou (*model*) od užívateľského rozhrania (*view*), vďaka čomu je možné každú časť modifikovať zvlášť a navyše je celá aplikácia jednotne a rozumne organizovaná. Takýto prístup implikuje menej komplikovaný a rýchlejší vývoj ako aj následné rozširovanie a spravovanie aplikácie. Model, View a Controller sú spolu prepojené a v neustálom kontakte, takže na seba navzájom odkazujú (*viď. Obrázok 2.1 - MVC štruktúra Zend Frameworku*) [13][14].



Obrázok 2.1 – MVC štruktúra Zend Frameworku

2.7 MySQL

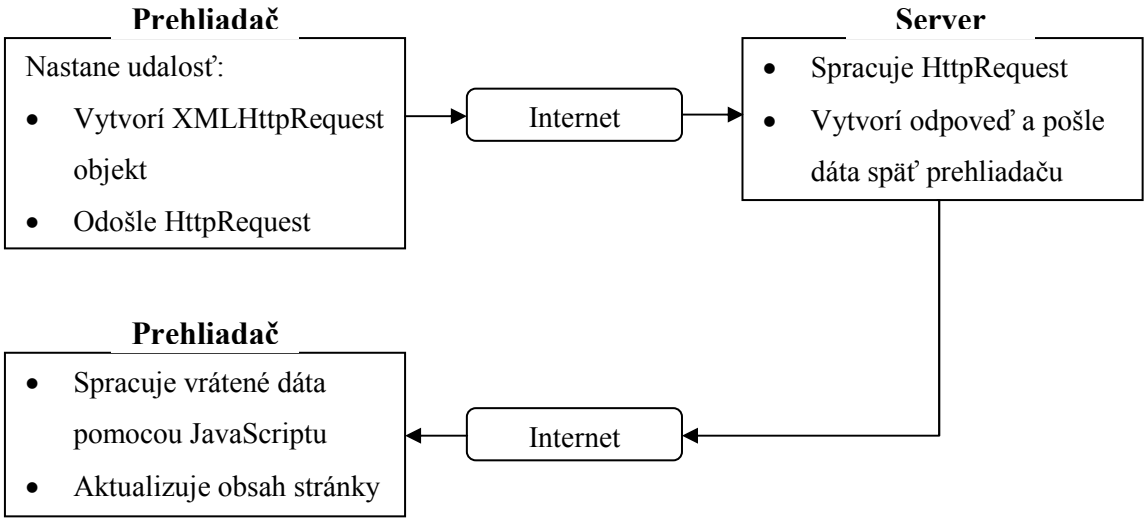
MySQL je SQL relačný databázový server. Je to najobľúbenejšia open-source databáza na svete vďaka svojej rýchlosti, vysokej spoľahlivosti a jednoduchej obsluhe. Z týchto dôvodov som si na uchovávanie dát pre tento projekt vybral práve túto databázu. Na jej lokálnu administráciu som používal *phpMyAdmin,* čo je open-source webové administrátorské rozhranie naprogramované v PHP. Neskôr na drobné úpravy už na online servri som používal program *SQLyog*.

2.8 AJAX

Tento projekt hudobného portálu je určený pre bežných ľudí a teda veľmi dôležitá je aj jeho prezentácia, užívateľské rozhranie a jeho pohodlné používanie. K tomu, aby sa užívateľ cítil na stránkach pohodlne, som využil technológiu AJAX.

AJAX znamená *Asynchronous JavaScript and XML.* Nie je to teda nový programovací jazyk, ale spôsob ako už existujúce jazyky a štandardy využívať. Časti aplikácie, ktoré využívajú AJAX, nie sú závislé na internetovom prehliadači ani platforme. AJAX umožňuje výmenu dát medzi klientom a serverom a aktualizovať tak časti obsahu stránky bez toho, aby bolo nutné stránku znovu načítať [15]. Používa na to v kombinácii:

- *XMLHttpRequest* objekt, aby vymenil data asynchrónne so serverom
- JavaScript/DOM, aby zobrazil a pracoval s informáciami
- CSS, aby dátam priradil štýly (vzhľad)
- *XML*, prípadne HTML alebo *JSON*, teda spôsob ako naformátuje odpoveď od serveru



Obrázok 2.2 – Ako pracuje AJAX

Na obrázku *Obrázok 2.2 – Ako pracuje AJAX* je po porade znázornený postup akcií pri AJAX-ovom volaní.

Rozhodol som sa AJAX použiť všade tam, kde by to mohlo užívateľovi spríjemniť ovládanie stránok, no aby celkový pocit ovládania nepôsobil zmätene a neprehľadne, ale naopak ešte viac intuitívne. Vďaka tomuto prístupu sú časté akcie vykonávané užívateľom pre neho menej otravné a tým pádom je pravdepodobnejšie, že od užívateľov portál získa viac informácií, vďaka čomu mu bude následne možné filtrovať stále presnejšie dáta.

Na formátovanie odpovedí od serveru som použil formát JSON.

2.9 JSON

JSON (*JavaScript Object Notation*) [16] je ľahko čitateľný formát na výmenu dát (primárne medzi serverom a webovou aplikáciou), ktorý som v aplikácií použil ako alternatívu k XML. Je založený na podmnožine JavaScriptu. Je to textový formát, ktorý je úplne nezávislý na programovacom jazyku a používa konvencie známe programátorom jazykov ako je C, C++, Java, JavaScript, Perl, Python a iné.

JSON odpoveď je objekt, ktorý sa skladá z neusporiadaného súboru párov v tvare *názov:hodnota.* Nakoľko je JSON podmnožina JavaScriptu, je jeho použitie v JavaScripte jednoduché, bez zbytočných komplikácií [16].

3 Analýza a ciele práce

Cieľom tejto bakalárskej práce je vytvorenie komunitného internetového hudobného portálu so zameraním na klubové akcie tanečných elektronických hudobných štýlov a jeho spustenie na online servri. To zahŕňa navrhnúť požiadavky zo strany

- **užívateľov**
  - vymyslieť využiteľné funkcie, kvôli ktorým má zmysel sa na stránku vrátiť
  - sprostredkovať intuitívne a efektívne užívateľské prostredie
  - klásť dôraz na bezpečnosť a rýchlosť
  - dať stránkam schopný dizajn
- **správcov**
  - ponúknuť prostriedky pre správu prvkov v databáze
  - dať možnosť udeľovania rôznych práv užívateľom
- **potenciálnych vývojárov**
  - písať dostatočne okomentovaný, prehľadný, dobre čitateľný a ľahko rozšíriteľný kód
  - dodržovať „zendovské“ programovacie zásady pre PHP

Cieľom bolo taktiež zvoliť pri návrhu a vývoji taký prístup, aby bol portál eventuálne v budúcnosti schopný existencie na čo najširšom území, a aby mohol fungovať s čo možno najmenším počtom zásahov od správcov a napriek tomu ponúkať stále nové a aktuálne informácie. To by malo mať za následok to, že by užívatelia dostali relatívne mocné práva a mohli by si tak vytvárať všetky podstatné informácie sami.

Dôležitá funkcionalita by mala spočívať vo filtrovaní takých hudobných akcií užívateľovi, ktoré by ho mohli zaujímať. Bolo teda treba vymyslieť systém, podľa ktorého bude aplikácia schopná rozhodovať o tom, či danú akciu užívateľovi odporučiť alebo nie.

3.1 Bezpečnosť

Bol kladený dôraz na bezpečnosť celej aplikácie. Bolo treba zabezpečiť účty užívateľov, ale aj samotnú aplikáciu.

3.1.1 Užívatelia

Užívateľom je v prvom rade nutné pri registrácií ukladať heslá do databázy v zašifrovanom tvare. Na tento účel som použil kryptovaciu funkciu ***SHA***, čiže *Secure Hash Algorithm* [17]. Jedná sa o skupinu kryptografických hašovacích funkcií, vyvinutú americkou Národnou Bezpečnostnou Agentúrou (NSA). Je to nástupca funkcie *MD5*. Z tejto skupiny som na zašifrovanie hesiel vybral funkciu *SHA-256,* ktorá používa 32-bitové slová. Pomocou PHP som potom jednoducho dané heslo zašifroval: `hash('sha256', $userData['password'])` [17].

Ďalšie dôležité nastavenie pre ochranu užívateľa je čas, za ktorý vyprší jeho *session*. Session sa obnovuje vždy, keď užívateľ prejde na inú podstránku. Ako náhle je užívateľ neaktívny dlhšie ako tento čas, session vyprší a užívateľ je z bezpečnostných dôvodov odhlásený zo systému. Tento čas v sekundách udáva parameter `session.gc_maxlifetime` a nastavuje sa v súbore *php.ini* na serveri. Na serveri je nastavený na 1440 sekúnd (24 minút). Aby to nebolo pre užívateľa príliš otravné, príkaz `session.cookie_lifetime` je nastavený na 0, čo znamená, že session vyprší až po zavretí prehliadača.

3.1.2 Aplikácia

Existujú rôzne útoky, ktoré je treba pri vytváraní webovej aplikácie mať na pamäti. Aby som potenciálnym útočníkom sťažil získanie informácií o aplikácií (aký používa software, akú má adresárovú štruktúru atp.), vypol som na serveri hlásenie chýb príkazom `error_reporting(0)`.

3.1.2.1 Validácia a filtrovanie vstupu

Je vhodné k vytváraniu takejto aplikácie pristupovať s myšlienkou „nikdy neveriť užívateľovi“. To znamená, že treba všetky jeho vstupy kontrolovať a validovať. Pri zadávaní informácií do konkrétneho formulára už popredu vieme, aké typy dát by mal užívateľ zadať. Preto napríklad vždy validujem formát času, dátumu, emailu, ale aj iných položiek, ktoré užívateľ zadáva. Tento spôsob validácie vstupu veľmi redukuje možnosť použitia akéhokoľvek *SQL útoku* [18].

V tejto aplikácií umožňujem užívateľovi si vybrať z vlastného disku a uploadnúť na server súbor. Buď ide o obrázok, ktorý bude užívateľa reprezentovať, alebo ak sa jedná o hudobnú akciu, tak ide o obrázok (plagát), ktorý reprezentuje danú akciu. V týchto miestach je dôležité kontrolovať, aký súbor sa užívateľ v skutočnosti snaží uploadnúť. V tomto prípade nebolo rozumné použiť tzv. „*black*

list“ a teda zakázať isté typy súborov, naopak omnoho vhodnejšie bolo zvoliť opačný prístup a povoliť len niektoré typy súborov [19]. Pomocou „zendovských“ validátorov som vymedzil užívateľovi možnosti uploadnúť len 1 súbor s príponou buď *jpg*, *jpeg*, *gif* alebo *png* o maximálnej veľkosti 1000kB.

3.1.2.2 XSS

Skratka *XSS* [20] znamená *Cross-Site Scripting*. Nastáva vtedy, keď aplikácia dostane od zlomyseľného užívateľa dáta, ktoré obsahujú JavaScript, VBScript, ActiveX, HTML, alebo Flash, aby takto napríklad oklamal užívateľa a tak získal o ňom informácie, prípadne poškodil vzhľad stránky. Preto som za každých okolností všetky vstupy od užívateľa (ale pre istotu aj všetky ostatné premenné) pri ich výstupe „eskejpoval“, čo znamená, že všetky špeciálne znaky sa pri výstupe skonvertujú na *HTML entity*. Použil som na to zendovskú metódu `escape()`, ktorá používa PHP funkciu `htmlspecialchars()` [10][12][20].

3.2 Optimalizácia rýchlosti

Aby návšteva stránky bola príjemná, je pre bežného užívateľa veľmi žiaduce, aby sa jednotlivé podstránky nenačítavali príliš dlho. Preto som sa snažil tento čas minimalizovať využitím rôznych techník a dodržiavaním istých zásad. Využil som na to taktiež informácie, ktoré mi poskytli programy *Yahoo! YSlow* [21] a *Google Page Speed* [22]. Jedná sa o zásuvné moduly do prehliadača Firefox, ktoré sú integrované do nástroja pre vývoj webových aplikácií *Firebug* [23].

YSlow a Page Speed analyzujú a známkujú webové stránky a navrhujú spôsoby ako zlepšiť ich výkon na základe rôznych pravidiel. Optimalizovať rýchlosť načítania stránok podľa niektorých pravidiel by bolo buď zbytočné, alebo nie úplne ideálne. Nasledujúce pravidlá sa mi podarilo dodržať úplne, alebo aspoň do istej miery.

3.2.1 Zredukovať počet HTTP požiadaviek

Podľa tohto pravidla pre skrátenie doby načítania stránok, je dôležité čo najviac zredukovať počet komponentov (tj. obrázky, súbory so štýlmi a skriptami, Flash atď.), ktoré musí klient stiahnuť. Aby som dosiahol čo najmenší počet HTTP požiadaviek, skombinoval som všetky kaskádové štýly do jedného súboru a všetky moje skripty do druhého. Toto pravidlo je veľmi podstatné, nakoľko zlepšuje dojem zo stránky pre užívateľov, ktorí ju navštívia prvýkrát, čo je pre úspech stránky jedno z kľúčových kritéri, pretože až 40-60% denných návštevníkov prichádza na stránky s prázdnou *cache* [21] [24].

3.2.2 Pridať hlavičky pre kontrolu vypršania platnosti

Na to, aby sa užívateľovi nemuseli komponenty sťahovať odznova vždy pri navštívení stránky, som využil príkazy, ktoré nastavia čas, ako dlho majú byť ktoré komponenty uchované v cache na klientskom počítači. Ide o tzv. *expires headers*. Tie ovplyvňujú počet HTTP požiadaviek pre užívateľov, ktorí stránku navštívili opakovane [21].

V súbore *.htaccess* v koreňovom adresári som tieto hlavičky nastavil pomocou príkazov:

```
ExpiresActive on
ExpiresDefault "access plus 24 hours"
ExpiresByType image/jpg "access plus 1 months"
```

Príkazom `ExpiresByType` som rovnaký čas vypršania platnosti jeden mesiac nastavil pre súbory typu jpg, ale aj pre súbory typu gif, png, css a skripty. To znamená, že všetky tieto komponenty sa budú ukladať v cache klienta po dobu jedného mesiaca.

3.2.3 Umiestniť CSS do značky head

Vďaka tomu, že sú štýly umiestnené už na vrchu dokumentu, môže byť stránka načítavaná postupne, takže prehliadač zobrazí akýkoľvek obsah hneď, ako náhle ho dostane. To dáva užívateľovi vizuálnu odozvu a ukazuje priebeh načítavania [21].

S postupným načítavaním stránky a CSS súvisí aj nastavovanie rozmerov obrázkom. Každému obrázku som pomocou CSS nadefinoval jeho šírku a výšku. Vďaka tomu, že prehliadač popredu pozná rozmery obrázkov, môže umiestňovať elementy okolo nich bez toho, aby bolo neskôr pri načítavaní nutné stránku alebo jej časť znova prekresľovať. Táto technika tiež urýchľuje vykreslenie celej stránky [22].

3.2.4 Umiestniť skripty na spodok dokumentu

Skripty spôsobujú problém, že blokujú paralelné sťahovanie iných komponentov. To znamená, že užívateľ bude nútený čakať dlhšie, kým si všimne akýkoľvek progres. Takže pokiaľ jediný účel JavaScript súboru je pridanie funkcionality (napríklad čo sa stane, keď je stlačené tlačidlo), je najlepšie takýto súbor umiestniť na spodok dokumentu, tesne pred uzatváraciu značku `</body>` [25].

Väčšinu funkcionality užívateľského rozhrania som umiestnil do jedného súboru a ten som umiestnil na koniec dokumentu. Niektoré skripty využívajúce jQuery mám v strede dokumentu priamo v `Zend_View`, pretože potrebujem ich časti vytvárať pomocou PHP a jeho premenných z daného kontroléru. Kvôli tomuto musia byť jQuery knižnice umiestnené stále na začiatku dokumentu, čo nie je úplne ideálne.

3.2.5 Komprimovať komponenty

Kompresia je jednoduchý a efektívny spôsob ako urýchliť načítavanie stránok. Niektoré veľmi staré prehliadače túto metódu nepodporujú, ale rozhodol som sa, že ju aj napriek tomu využijem. Používajú ju aj stránky ako Google alebo Yahoo. Aby kompresia fungovala, musí prehliadač akceptovať komprimovaný obsah. Na opačnej strane je treba nakonfigurovať server, aby takýto komprimovaný obsah vracal v prípade, ak s ním prehliadač vie zaobchádzať [26] . To som dosiahol týmito príkazmi, ktoré som pridal do .htaccess súboru:

```
FilterProvider gzipping deflate resp=Content-Type text/html
FilterProvider gzipping deflate resp=Content-Type text/css
FilterProvider gzipping deflate resp=Content-Type $javascript
```

Komprimujem len html súbory, štýly a skripty. Nie je žiaduce komprimovať obrázky, nakoľko sú už skomprimované. Malo by to za následok zbytočné zaťaženie procesoru, prípadne zväčšenie ich veľkosti. Po prihlásení sa do tohto portálu prehliadač stiahol 691KB. Komprimovanie komponentov toto číslo zredukovalo na 363KB, čím sa zlepšil čas načítania stránky.

3.2.6 Optimalizácia jQuery

Na viacerých miestach v aplikácií som využil knižnicu jQuery. Táto knižnica veľmi spríjemňuje prácu vývojára, no treba ju používať správne, pretože inak môže naopak veľmi znepríjemniť užívateľovu návštevu stránky, ktorá by mohla byť zbytočne príliš pomalá. V tejto časti opíšem niekoľko praktík, ktoré som sa snažil využívať, aby funkcionalita spojená s touto knižnicou pracovala čo najefektívnejšie.

Načítať knižnice pomocou *Google AJAX Libraries API* [27] Jedná sa o službu poskytovanú spoločnosťou Google, ktorá umožňuje načítavať najobľúbenejšie open-source JavaScript knižnice priamo z ich stránok, namiesto z vlastného servera. Pomocou tejto služby sa knižnice stiahnu veľmi rýchlo. Navyše ak užívateľ už pred tým navštívil nejakú stránku, ktorá rovnaké knižnice tiež sťahuje pomocou tejto služby, sú potrebné súbory už uložené v cache. Týmto spôsobom sa sťahujú dva najväčšie súbory v komprimovanom tvare, ak to prehliadač umožňuje: knižnice jQuery 1.4.2 a jQuery UI 1.8.1.

Nevyberať ten istý element viackrát Pomocou jQuery selektorov sa dá vybrať akýkoľvek element z DOM. Je však nevýhodne zaťažovať procesor znova a znova tými istými selektormi, pretože interne musia spraviť značný kus práce, aby daný element našli. Je teda rozumné si nájdené elementy uložiť do pomocných premenných a následne využívať tie [28].

Vyberať elementy pomocou identifikátoru [28] Selektory jQuery umožňujú vyberať elementy viacerými spôsobmi. Napríklad pomocou triedy, alebo identifikátoru elementu. Vždy keď je to možné, je omnoho lepšie zadať selektoru identifikátor elementu. jQuery potom na nájdenie elementu využíva natívnu metódu JavaScriptu `getElementById` a nemusí na to používať žiadne svoje prechádzanie dokumentu, čo je omnoho rýchlejšie.

Zadávať selektorom kontext [28] Ak zadáme selektoru element, ktorý má hľadať, bude prechádzaný celý dokument, kým sa element nenájde. jQuery umožňuje pridať druhý parameter tzv. *kontext*. Keď selektoru určíme kontext, znamená to, že sme selektoru ukázali element, v ktorom má začať hľadať, takže nemusí prechádzať celú štruktúru dokumentu.

3.3 SEO

SEO [29] je skratka pre *Search Engine Optimalization*, čo je optimalizácia pre internetové vyhľadávače ako je Google. Je to proces, pomocou ktorého je možné zlepšiť umiestnenie stránky v zozname nájdených výsledkov vyhľadávačov a tak potenciálne zvýšiť počet návštevníkov stránky. Google odporúča používať celú radu techník, aby vyhľadávače mohli jednoduchšie *zaindexovať* obsah na danej stránke, aby užívatelia mali lepší dojem zo stránky a aby sa stránka umiestnila na lepšej pozícií medzi vyhľadanými výsledkami [29]. Nasleduje niekoľko vybraných odporúčaní, podľa ktorých som sa pri vývoji tohto projektu riadil.

Vytvárať jedinečné a presné názvy stránok [29] Značka *title* prezradí používateľom aj vyhľadávacím nástrojom tému konkrétnej stránky. V ideálnom prípade by každá stránka mala mať jedinečný názov. Názvy by mali byť krátke a zároveň informatívne.

Využívať metaznačku *description* [29] Metaznačka *description* stránky poskytuje vyhľadávaču Google a iným vyhľadávacím nástrojom súhrnné informácie o obsahu stránky. Je dôležitá, pretože ju Google môže použiť ako úryvok pre danú stránku.

Štruktúra URL adries [29] Vytvorenie popisných kategórií a názvov súborov pre dokumenty na stránkach môže pomôcť udržať prehľadnosť stránok. Vyhľadávacím nástrojom môže zase uľahčiť indexové prehľadávanie dokumentov na stránke. Adresy URL so slovami, relevantnými pre obsah a štruktúru stránok, sú priateľskejšie pre užívateľov, ktorí stránkami prechádzajú. Návštevníci si ich ľahšie zapamätajú a môžu byť ochotnejší odkazovať na ne. Vďaka Zend Framework majú stránky tohto portálu „používateľsky prístupné“ URL adresy ako napríklad: http://musicportal/home/event/add/.

Používať správne značky nadpisov [29] Účelom značiek nadpisov je popísať štruktúru stránky pre používateľov. Správne použitie viacerých veľkostí nadpisu vytvára pre obsah hierarchickú štruktúru a používateľom tak zjednoduší prechádzanie dokumentom. Mojím cieľom bolo používať aj ostatné HTML značky tak, aby čo najlepšie popisovali dokument a udržovali tak jeho správnu sémantiku.

Optimalizovať prácu s obrázkami [29] Každý obrázok môže mať odlišný názov súboru a atribút *alt*. V prípade, že sa obrázok z akéhokoľvek dôvodu nezobrazuje, atribút alt umožní určiť pre neho alternatívny text. Je vhodné používať stručné, ale popisné názvy súborov aj alternatívneho textu. Vyplnenie alternatívneho textu obrázku pomôže prehľadávaču Google lepšie rozpoznať zameranie stránky, na ktorú obrázok odkazuje.

4 Návrh

Začiatku práce na každom celku predchádzala dôkladná analýza problému a návrh čo možno najlepšieho riešenia z hľadiska používateľa, ale aj programátora. Bolo sa treba zamyslieť nad tým, kto bude stránky navštevovať, pre koho sú určené. Malo by ísť prevažne o mladých ľudí so záľubou v elektronickej hudbe. Podľa toho bol navrhnutý moderný dizajn a užívateľské prostredie. Ďalej bolo treba navrhnúť aplikačnú logiku, databázový model, právomoci užívateľov a systém, podľa ktorého sa bude filtrovať obsah pre konkrétneho používateľa.

4.1 Diagram prípadov použitia

Diagram prípadov použitia (angl. use case diagram) je typ diagramu v jazyku UML. UML nedefinuje štandardy zápisu prípadov použitia, grafické zobrazenie poskytuje iba najzákladnejší prehľad prípadov použitia alebo sád prípadov použitia. Každý prípad použitia opisuje jeden spôsob (dôvod) použitia systému z hľadiska používateľa. Súbor všetkých prípadov použitia potom reprezentuje všetky používateľské funkcie, ktoré budúci systém ponúkne [30].

Portál je navrhnutý takým spôsobom, aby sa na ňom objavovalo čo najviac nových informácií. Malo by to byť dosiahnuté vďaka tomu, že každý **registrovaný užívateľ** má práva pridávať viacero vecí, medzi ktoré patria najmä najdôležitejšie hudobné akcie (v rámci kódu nazvané `events`) a hudobné sety (`sets`), čo sú základné prvky portálu. V pôvodnom návrhu sú ešte iné položky, ktoré by mohol užívateľ pridávať, no v tejto iterácii nie sú implementované (*viď. Kapitola 7 – Možné vylepšenia*). **Správca** môže navyše pridávať interpretov (`artists`) a kluby (`clubs`). Názornejší je diagram prípadov použitia, ktorý ukazuje aké majú jednotliví účastníci možnosti (*viď. Obrázok 4.1 – Diagram prípadov použitia*).

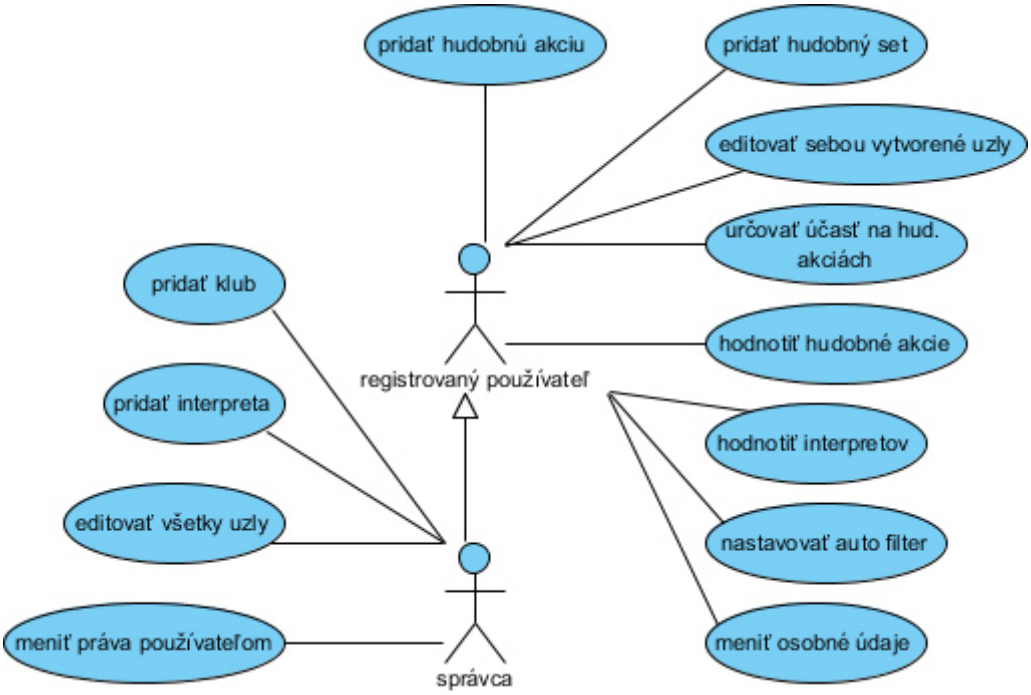
4.2 Databázový návrh a ER diagram

Databázové tabuľky boli navrhnuté tak, že všetky príspevky, ktoré užívatelia môžu pridávať (interpreti, kluby, hudobné akcie a sety) sa skladajú z dvoch častí, ktoré tvoria jeden celok. Vždy, keď užívateľ niečo pridá, vytvorí sa v databáze minimálne dvojica nových riadkov. Prvý riadok sa vyrobí v tabuľke *nodes* (*uzly*), ktorá obsahuje základné informácie ako je čas vytvorenia uzlu, identifikátor užívateľa, ktorý ho vytvoril a identifikátor typu daného uzlu. Na základe typu uzlu sa vyberie druhá tabuľka, kde sa uložia konkrétnejšie informácie o danom príspevku spolu s identifikátorom *id_node*, ktorý odkazuje na príslušný rodičovský riadok v tabuľke *nodes*. Takže keď používateľ pridá napríklad hudobný set, vytvorí sa riadok pre uzol v tabuľke *nodes* a riadok hudobného setu v tabuľke *sets*, ktorý naň odkazuje.

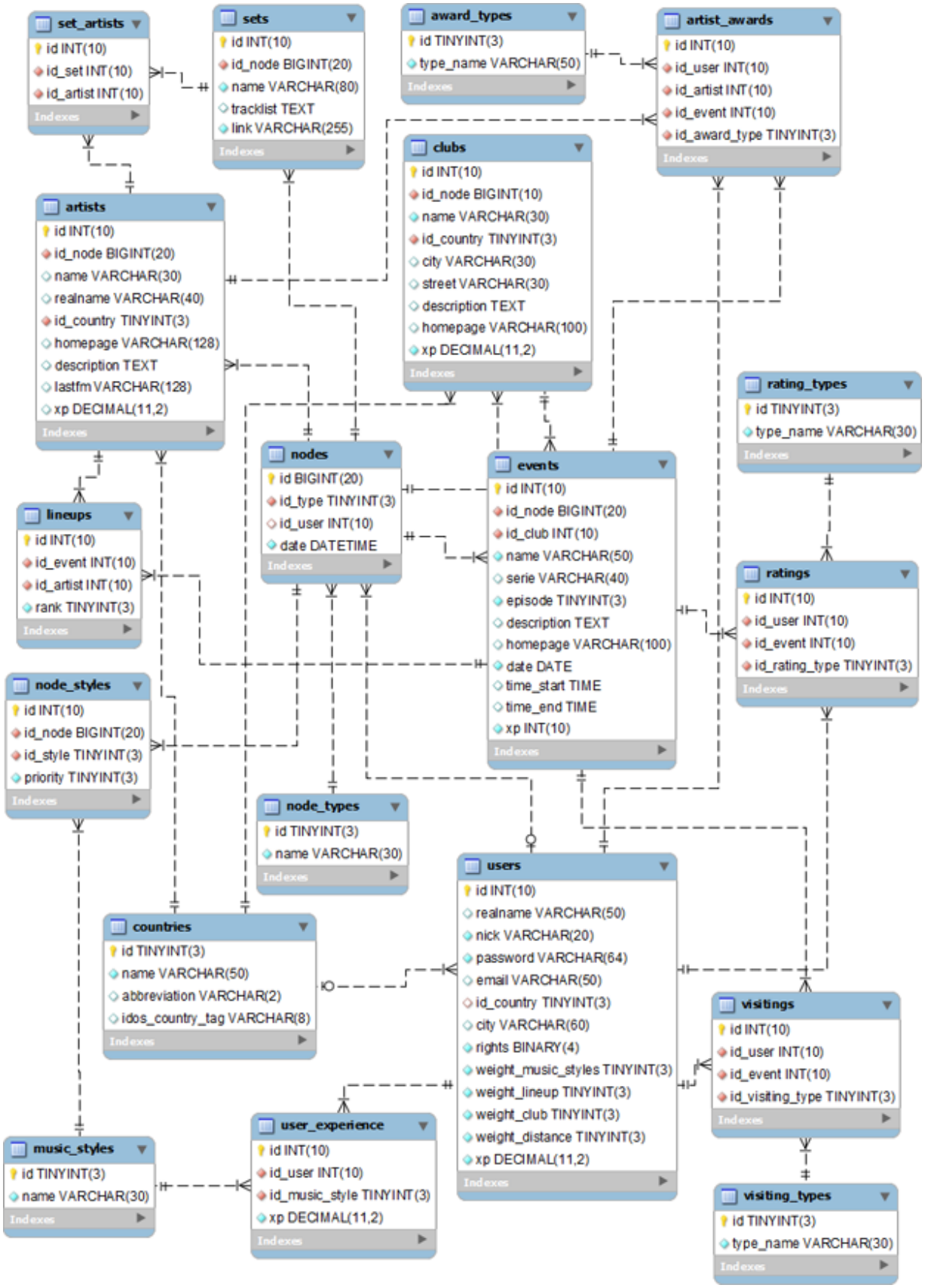
Ak používateľ pridá novú hudobnú akciu, na ktorej vystupujú interpreti, ktorí ešte nie sú uložení v databáze, vytvoria sa pre nich nové uzly a im prislúchajúce riadky v tabuľke *artists*. To isté platí aj pre neexistujúci klub, ak sa v takom má konať nová hudobná akcia.

Databázový model znázorňuje ER diagram (*viď. Obrázok 4.2 – ER diagram*). Na jeho návrh som použil program *MySQL Workbench*. Tabuľky sú typu *InnoDB*, aby bolo možné využiť tzv. *foreign keys* (cudzie kľúče). Pomocou týchto cudzích kľúčov sú tabuľky poprepájane vzťahmi s nastavením tzv. *kaskádového mazania* tam, kde to dáva zmysel. Takže pri vymazaní riadku z tabuľky *nodes*, sa vymažú všetky napojené riadky z iných tabuliek, ale napríklad riadok krajiny z tabuľky *countries*, alebo riadok pre hudobný štýl z tabuľky *music_styles* ostane zachovaný napriek tomu, že sa na neho z nejakej tabuľky odkazovalo.

Dátové typy jednotlivých položiek sú vyberané „ekonomicky“ aby sa zbytočne neplytvalo miestom na pevnom disku. Pre identifikačné čísla krajín, typov uzlov, hudobných štýlov a podobných tabuliek s malým počtom riadkov je použitý typ `unsigned tinyint`. Naopak tabuľka *nodes* má identifikačné číslo typu `unsigned bigint`, pretože obsahuje najviac položiek. Išlo o to, pripraviť aplikáciu do budúcnosti, kedy by tento typ mohol byť potenciálne využitý. Maximálne dĺžky názvov známych vecí ako sú mestá, štáty, mnou vymyslené názvy (napr. typov ocenení) a podobných položiek sú vybrané podľa najdlhšieho možného, aké existuje. Všetky tabuľky a položky v nich majú *collation* nastavenú na *utf8_unicode_ci*, z dôvodu teoretického medzinárodného využitia portálu v budúcnosti.

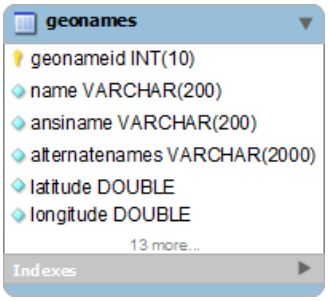


Obrázok 4.1 – Diagram prípadov použitia



Obrázok 4.2 – ER diagram

Databáza obsahuje ešte jednu tabuľku, ktorá nemá žiadne vzťahy. Je preto do tejto správy vložená zvlášť. Jedná sa o tabuľku *geonames* (*viď. Obrázok 4.3 – Tabuľka geonames*).



Obrázok 4.3 – Tabuľka geonames

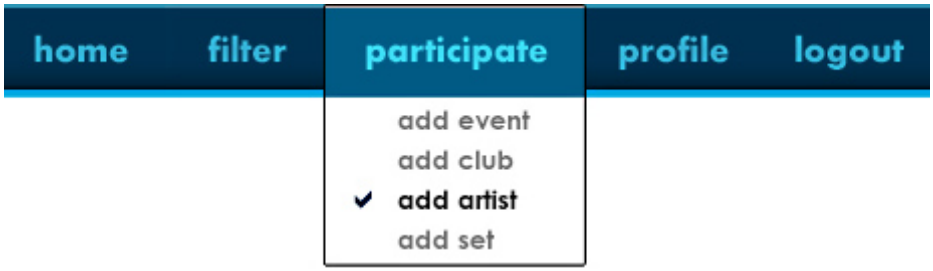
Popis jednotlivých tabuliek ER diagramu:

- nodes – ako bolo povedané, ide o základnú tabuľku, na ktorú sa odkazujú tabuľky *artists*, *events*, *clubs* a *sets*. Sú to uzly, ktoré môžu užívatelia pridávať do systému.
- node_types – uzol je vždy jedného konkrétneho typu. Tie sú uložené v tejto tabuľke.
- node_styles – V tejto tabuľke sú uložené identifikátory hudobných štýlov, ktoré sa môžu k uzlom vzťahovať (napr. k akciám alebo k setom). Niektoré štýly môžu dôležitosťou prevažovať ostatné. Na to slúži stĺpec *priority*. Najväčšia priorita je „3“.
- music_styles – tu sú uložené jednotlivé hudobné štýly.
- events – tabuľka pre uchovávanie hudobných akcií. Obsahuje pre používateľov dôležité informácie o akciách, ako napr. dátum a čas danej akcie. Stĺpec *id_club* odkazuje na príslušný klub, v ktorom sa akcia koná. Stĺpec *xp* (skratka z anglického slova experience, teda skúsenosti) obsahuje počet „xp bodov“, ktoré sa použijú, keď používateľ udeľuje ceny za najlepšieho účinkujúceho na danej hudobnej akcii (*viď. časť 4.4 Automatický filter*).
- lineups – na hudobnej akcií väčšinou vystupuje viacero interpretov. Táto tabuľka slúži na uloženie ich identifikátorov s odkazom na konkrétnu akciu, ku ktorej patria. Stĺpec *rank* určuje dôležitosť jednotlivých interpretov, nakoľko je pravidlom, že na akcií je vždy jeden alebo viac hlavných dídžejov tzv. „headlinerov“ a ostatní ich dopĺňajú. Rank s hodnotou „1“ značí headlinera, rank „2“ značí doplňujúceho dídžeja.
- clubs – v tejto tabuľke sú uložené informácie o kluboch. Stĺpec *id_country* odkazuje na konkrétnu krajinu z tabuľky *countries*. V stĺpci *xp* je uchovaný počet xp bodov, ktoré daný klub získal.
- artists – tu sú uložení jednotliví interpreti. Tí majú podobne ako kluby odkaz na krajinu odkiaľ pochádzajú a stĺpec pre xp body, ktoré získali.

- countries – tabuľka so všetkými krajinami na Zemi. Pri každej je uložená ich dvojmiestna skratka podľa štandardu ISO 3166 [31] a druhá skratka, kvôli službe vyhľadávania spojov *idos.cz* [32], ktorá používa iné skratky pre krajiny.
- artist_awards – užívateľ môže pri každej akcií, ktorá sa konala, udeliť dídžejom rôzne ocenenia. Tie sa ukladajú do tejto tabuľky.
- award_types – typy udelených ocenení (implementované len jedno ocenenie za najlepšieho dídžeja).
- sets – jednoduchá tabuľka pre ukladanie hudobných setov.
- set_artists - hudobný set mohlo „namixovať“ prípadne viac dídžejov. Na ich ukladanie slúži táto tabuľka.
- users – informácie o každom užívateľovi, ktorý sa zaregistruje, sa uložia do tabuľky *users*. Užívateľove práva (stĺpec *rights*) sa implicitne nastavia na binárny reťazec „11000“. Sú to základné práva, ktoré sú znázornené v diagrame prípadov použitia. Prvá jednotka vraví, že má užívateľ právo pridávať hudobné akcie. Druhá jednotka reprezentuje sety. Nasledujúca 0 značí, že užívateľ nemá právo pridávať kluby. Ďalšia cifra hovorí o pridávaní interpretov a posledná, najpravejšia cifra zahrňuje ostatné práva tj. možnosť editácie všetkých uzlov a prideľovanie práv ostatným používateľom. Správcovia majú vždy práva nastavené na „11111“. Stĺpce *weight_music_styles*, *weight_lineup*, *weight_club* a *weight_distance* udávajú váhu (dôležitosť) kritérií pri filtrovaní obsahu pomocou automatického filtru. Pri registrácii sú všetky váhy implicitne nastavené na „3“, čo je stredná váha.
- visitings – používateľ môže pri každej hudobnej akcií rozhodnúť o tom, či sa jej zúčastní. Typy účasti sú uložené v tabuľke *visiting_types*. Tabuľka *visitings* ukladá to, ktorý užívateľ má pri ktorej akcii ktorý typ účasti.
- visiting_types – tu sú uložené typy účasti. Konkrétne typy v preklade znamenajú: „zúčastním sa“, „nie som si ešte istý“, „nemôžem alebo nechcem sa zúčastniť“.
- ratings – k akciám, ktoré sa už konali, môže užívateľ (podobne ako účasť) vybrať svoje hodnotenie danej akcie.
- rating_types – typy hodnotení sú uložené v tejto tabuľke. V preklade znamenajú: „bavil som sa“, „priemerná akcia“, „nepáčila sa mi“ a „nezúčastnil som sa“.
- user_experience –užívateľom sa taktiež ukladajú xp body. Na rozdiel od klubov a interpretov sa im tieto xp body delia podľa hudobných štýlov. Aby nebolo treba vždy po pridaní nového štýlu vytvárať nový stĺpec v tabuľke *users*, rozhodol som sa xp body užívateľom ukladať do tejto špeciálnej tabuľky.
- geonames – tabuľka, ktorá obsahuje miesta na Zemi (mestá, dediny) a ich príslušné zemepisné šírky a dĺžky. Obsahuje ešte ďalšie stĺpce, ktoré však aplikácia nevyužíva.

4.3 Užívateľské rozhranie

Pri vytváraní užívateľského rozhrania bol kladený dôraz na to, aby bol praktický, jednoduchý a prehľadný. Išlo o to, aby bolo používateľovi všetko hneď jasné a rozhranie ho nenútilo príliš rozmýšľať. Na úvodnej stránke sa nachádza len logo, jednoduchý formulár pre prihlásenie sa a tlačidlo pre registráciu. Po prihlásení sa namiesto tlačidla pre registráciu objaví menu s tlačidlami pre domovskú stránku, filtrovanie hudobných akcií, pridávanie obsahu, vlastný profil a odhlásenie zo systému. Tlačidlá pre filtrovanie akcií a pridávanie obsahu po prejdení myšou zobrazujú submenu s ďalším výberom (*viď. Obrázok 4.4 – Hlavné menu a submenu pre pridávanie obsahu*).



Obrázok 4.4 – Hlavné menu a submenu pre pridávanie obsahu

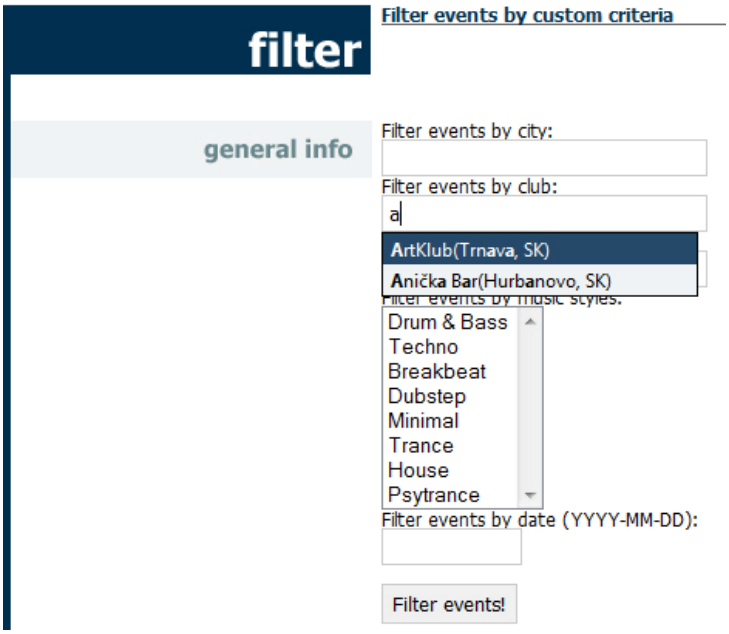
Na „**domácej stránke**“ (*home*) sa užívateľovi zobrazuje zoznam hudobných akcií, ktorým určil svoju účasť. Pod nimi sa zobrazujú akcie, na ktorých sa zúčastnil, teda ktorým vybral hodnotenie. Ak ale vybral danej akcii možnosť, že sa jej v skutočnosti nezúčastnil, prestane sa mu v zozname zobrazovať.

Keď nastane situácia, že sa daná hudobná akcia už konala a užívateľ mal pri nej vybratú účasť buď typu, že sa *zúčastní*, alebo sa *možno zúčastní*, zobrazí sa táto akcia na vrchu, aby užívateľa na seba upozornila. Tam sa bude dovtedy, kým ju užívateľ neohodnotí. Pri týchto akciách, ktoré sa už konali, môže užívateľ udeľovať ceny účinkujúcim dídžejom. Po ocenení dídžeja sa pri ňom objaví malá „bublinka“, ktorá oznámi, že za to daný dídžej dostal xp body. Ďalšia podobná bublinka sa objaví v pravom dolnom rohu obrazovky, ktorá oznamuje, že xp body dostal aj sám užívateľ (*viď. Obrázok 4.5 - Udelenie ceny najlepšiemu interpretovi*). Malo by to užívateľov motivovať k tomu, aby túto možnosť naozaj využívali, pretože to akýmsi spôsobom „pomáha“ ich obľúbenému interpretovi, ale aj užívateľom samotným. Užívateľom to pomáha konkrétne tak, že im vďaka tomu bude systém schopný filtrovať stále vhodnejší obsah.

Na obrázku *Obrázok 4.5 – Udelenie ceny najlepšiemu interpretovi* je vidno aj menu danej hudobnej akcie. To sa zobrazuje len vtedy, ak sa nad konkrétnou akciou nachádza kurzor. Vďaka tomu je stránka menej preplnená, viac dynamická a teda atraktívnejšia pre užívateľa. Budúce akcie majú v menu navyše možnosť vyhľadania spojenia pomocou služby idos.cz (*viď. Časť 5.6 Vyhľadávanie IDOS*).

Obrázok 4.5 – Udelenie ceny najlepšiemu interpretovi

Tlačidlo **filter** zobrazuje submenu, ktoré dáva užívateľovi na výber medzi **automatickým** a **manuálnym** filtrovaním. Manuálne filtrovanie ponúka možnosť zvoliť kritéria, podľa ktorých sa vyfiltrujú hudobné akcie v systéme. Užívateľ môže určiť nula (zobrazia sa všetky akcie v systéme) až všetky kritéria, ktoré vyfiltrované akcie musia všetky splňovať. Je možné filtrovať podľa mesta, klubu, názvu akcie, podľa hudobných štýlov alebo dátumu. Pri názve klubu a názve hudobnej akcie systém navrhuje už existujúce prvky v systéme (*viď. Obrázok 4.6 – Automatické dopĺnanie textového poľa*). Tejto funkcionality je dosiahnuté pomocou jQuery pluginu *autocomplete* a je použitá pri všetkých formulárových poliach v aplikácii, kde sa očakáva na vstupe klub alebo interpret.



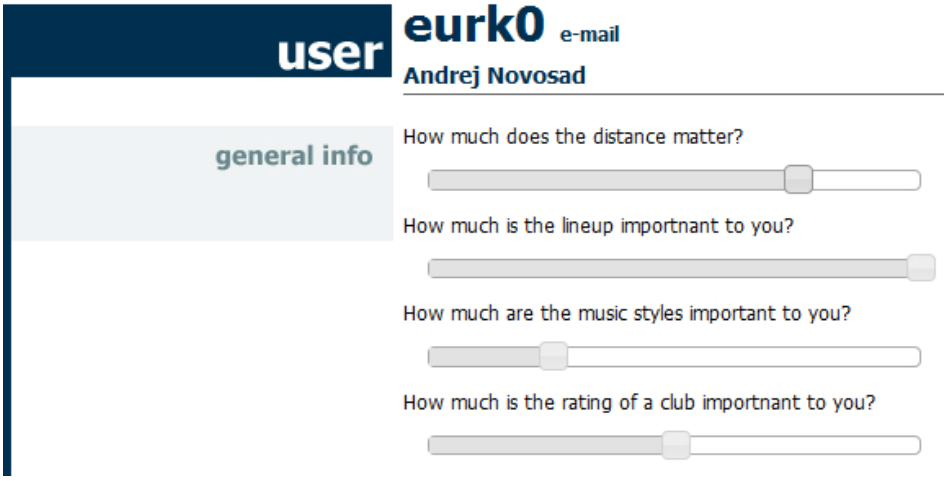
Obrázok 4.6 – Automatické dopĺňanie textového poľa

Automatický filter filtruje budúce akcie podľa toho, ako má užívateľ nastavené váhy jednotlivých kritérií tj. ako sú pre neho jednotlivé kritéria podstatné:

- **vzdialenosť mesta**, v ktorom sa akcia odohráva od mesta, kde býva užívateľ

- **váha „lineupu“ akcie** (lineup je spoločný názov pre všetkých vystupujúcich interpretov na danej hudobnej akcii)
- **váha hudobných štýlov akcie**
- **váha hodnotenia klubu**, v ktorom sa akcia odohráva

Tieto váhy si užívateľ môže nastaviť vo svojom **profile** pomocou posuvníkov, na ktorých vytvorenie je použitý plugin *Slider* z knižnice jQuery UI (*viď. Obrázok 4.7 – Nastavenie automatického filtra*). Ich nastavenie nie je treba nijako potvrdzovať, zmeny sa ukladajú okamžite pomocou technológie AJAX.



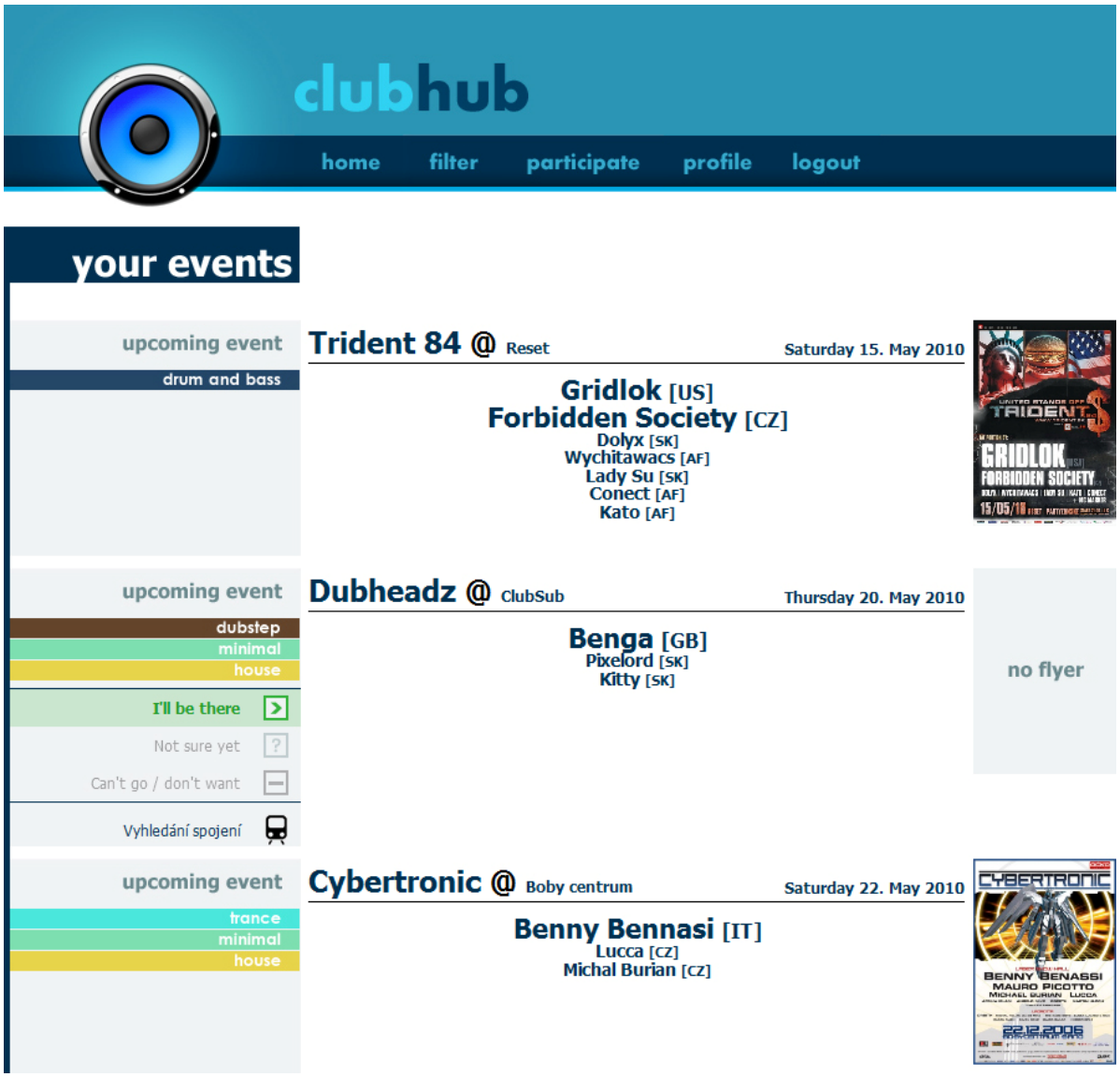
Obrázok 4.7 – Nastavenie automatického filtru

Udeľovanie práv je spravené veľmi jednoducho pomocou zaškrtávacích políčok, čo je pre aplikáciu úplne dostačujúce. Správca klikne na ikonku užívateľa, kde mu môže zrušiť alebo pridať:

- práva pre pridávanie hudobných akcií
- práva pre pridávanie hudobných setov
- práva pre pridávanie klubov
- práva pre pridávanie interpretov
- správcovské práva

Práva sa užívateľovi aktualizujú a uložia ihneď po kliknutí, bez nutnosti čokoľvek potvrdzovať.

Dizajn užívateľského rozhrania bol vytvorený pomocou programu *Adobe Photoshop CS3*. Jeho celkový vzhľad je ukázaný na obrázku *Obrázok 4.8 – Ukážka užívateľského rozhrania*, kde je zobrazený príklad domácej stránky (dostupnej pomocou tlačidla *home*).



Obrázok 4.8 – Ukážka užívateľského rozhrania

4.4 Automatický filter

V tejto časti je vysvetlený princíp systému, pomocou ktorého sa aplikácia postupne učí, čo sa danému užívateľovi páči a podľa toho mu odporúča obsah. Aby mohol fungovať spomínaný automatický filter, bolo treba navrhnúť spôsob, akým sa bude rozlišovať, ktoré akcie sú pre daného užívateľa lepšie, a ktoré horšie.

Kvalitu hudobnej akcie určujú viaceré faktory. Pre užívateľa, ktorý sa rozhoduje, či na nejakú akciu pôjde alebo nie, je dôležitá obec, kde sa daná akcia koná, resp. **vzdialenosť** tejto obce od obce, kde užívateľ sám býva. Táto vzdialenosť je prvé a najpodstatnejšie kritérium, ktoré automatický filter využíva.

Fanúšikovia tanečnej elektronickej hudby počúvajú často viaceré štýly, no vo väčšine prípadov jeden štýl, ktorý majú najradšej, prevažuje ostatné. Niektorí dokonca obľubujú len jeden štýl

a navštevujú akcie zamerané len na neho. Preto je pre užívateľa veľmi dôležité, aké **hudobné štýly** sa na danej akcii majú hrať, čo je druhé najpodstatnejšie kritérium, ktoré zohľadňuje automatický filter.

Na akciu spravidla chodí viac ľudí, keď na nej vystupujú známi a kvalitní interpreti. **Zoznam účinkujúcich** (angl. lineup) je tretie kritérium, ktorým sa tento filter riadi.

Posledným, najmenej podstatným kritériom, je **hodnotenie klubu**. Kvalita klubu (tj. či v ňom vystupujú kvalitní a obľúbení dídžeji a či sa ľudom páčia akcie, ktoré sa v ňom konajú), je tiež určite podstatná, no z vlastných skúseností ale aj zo skúseností iných ľudí, ktorí sa v tejto oblasti pohybujú, som zadal tomuto kritériu najmenšiu váhu. Váhy sa ustálili na konkrétnych hodnotách po niekoľkonásobnom testovaní. Máme teda 4 kritéria pre filtrovanie hudobných akcií zoradené podľa závažnosti:

1. **vzdialenosť klubu** má vo výslednom výpočte váhu 7
2. **hudobné štýly** majú váhu 5
3. **kvalita lineupu** má váhu 4
4. **kvalita klubu** má váhu 3

Automatický filter ich používa k tomu, aby vybral pre užívateľa len tie akcie, ktoré by ho mohli najviac zaujímať. Nasledujúca otázka znela, akým spôsobom určovať jednotlivé kritéria.

4.4.1 Vzdialenosť klubu

Ak sa akcia koná v klube, ktorý je v inom meste, než býva užívateľ, je treba vypočítať vzdialenosť tohto mesta od užívateľovho mesta. Na výpočet vzdialenosti medzi mestami som zvolil vzorec (*viď. Vzorec 4.1*) na výpočet najkratšej vzdialenosti medzi ktorýmikoľvek bodmi na povrchu gule, ktorá je meraná po povrchu, nie cez vnútro gule (angl. *Great Circle Distance*) [33]. Planéta Zem síce nie je dokonalá guľa, no odchýlka je zanedbateľná a tento vzorec sa napriek tomu využíva pre navigáciu v rôznych oblastiach.

Ktorékoľvek dva body na povrchu gule, ktoré neležia presne oproti sebe, ležia práve na jednom unikátnom *veľkom kruhu.* Veľký kruh je kruh na povrchu gule, ktorý má stred v rovnakom bode, ako guľa [33].

$$\Delta\hat{\sigma} = \arccos(\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos\Delta\lambda) \tag{4.1}$$

Do vzorcu sa dosádzajú zemepisné šírky (ϕ) a zemepisné dĺžky (λ) oboch miest, čím sa získa *centrálny uhol.* Vzdialenosť daných bodov pre guľu s polomerom r a $\Delta\hat{\sigma}$ daným v radiánoch je potom:

$$d = r\Delta\hat{\sigma} \tag{4.2}$$

Zemepisné šírky a dĺžky okrem iného zbiera a zdarma ponúka služba **GeoNames** [34]. Ide o geografickú databázu pokrývajúcu všetky krajiny a obsahujúcu viac ako 8 miliónov miest, medzi

ktorými sú aj tie najmenšie obce. Odtiaľ som stiahol a do mojej databázy vložil všetky obce z Českej a Slovenskej Republiky. Zemepisné šírky a dĺžky nutné pred vložením do vzorca previesť na radiány. Na to slúži funkcia jayka PHP `deg2rad`. Pseudokód pre výpočet najkratšej vzdialenosti medzi obcami s využitím zemepisných šírok a dĺžok teda vyzerá nasledovne:

```
vzdialenost = arccos(sin(deg2rad(miesto1_zemSirka))
                   * sin(deg2rad(miesto2_zemSirka))
                   + cos(deg2rad(miesto1_zemSirka))
                   * cos(deg2rad(miesto2_zemSirka))
                   * cos(deg2rad(miesto1_zemDlzka - miesto2_zemDlzka)));

vzdialenost = rad2deg(vzdialenost);
mile = vzdialenost * 69;
km = mile * 1.61;
```

Získanú vzdialenosť v km je treba ohodnotiť. Každé zo štyroch kritérií je obodované a akcia môže zaň získať maximálne 10 bodov. Na základné ohodnotenie vzdialenosti som podľa vlastného uváženia a po viacerých testoch prišiel k takýmto bodovým hodnotám:

- **10 bodov** ak sa akcia koná v rovnakom meste, v akom býva užívateľ (v tomto prípade sa vzdialenosť samozrejme zbytočne nepočíta)
- **9 bodov** ak je vzdialenosť menšia ako 15km
- **8 bodov** ak je vzdialenosť menšia ako 30km
- **7 bodov** ak je vzdialenosť menšia ako 50km
- **6 bodov** ak je vzdialenosť menšia ako 80km
- **5 bodov** ak je vzdialenosť menšia ako 100km
- **4 body** ak je vzdialenosť menšia ako 150km
- **3 body** ak je vzdialenosť menšia ako 200km
- **2 body** ak je vzdialenosť menšia ako 250km
- **1 bod** ak je vzdialenosť menšia ako 300km
- **0 bodov** ak je vzdialenosť menšia ako 400km

Ak je vzdialenosť ešte väčšia, body pokračujú postupne do mínusu. To preto, aby sa akcie, ktoré sú síce vynikajúce podľa všetkých ostatných kritérií, aj tak nezobrazovali, pretože sa konajú naozaj príliš ďaleko.

4.4.2 Kvalita účinkujúcich

Na to, aby systém mohol obodovať účinkujúcich (lineup), musí mať o jednotlivých interpretoch, z ktorého sa lineup skladá, nejaké podstatné informácie z hľadiska ich kvality.

Účinkujúci (*dídžeji z angl. slova DJ, teda disc jockey*) môžu niekoľkými spôsobmi získavať už spomínané xp body (*viď. Časť 4.4.6 Spôsoby získavania xp bodov*). Ak dídžej dosiahne počet xp bodov istej hranice, zväčší sa jeho „úroveň“ (*angl.* ***level***). V systéme obecne platí, že čím vyšší level,

tým je to pre daný subjekt, či už ide o dídžeja, užívateľa alebo klub, lepšie, pretože filtrer dokáže akcie presnejšie filtrovať vďaka zúčastneným vyšším levelom. Počet xp bodov *x*, ktoré dídžej musí dosiahnuť pre level *n* udáva vzorec 4.3:

$$x = 3 * (3 * n + 2) * (n - 1) \tag{4.3}$$

Body za lineup sa počítajú pomocou levelov jednotlivých dídžejov, z ktorých sa lineup danej hudobnej akcie skladá. Na akciu chodia ľudia najmä kvôli tzv. *headlinerom*. To sú hlavní účinkujúci, ktorých užívateľ uložil do databázy s rankom 1 cez textové pole *headliners*, keď pridal konkrétnu hudobnú akciu. Zvyšní účinkujúci ich len dopĺňajú. Headlineri teda majú vo výpočte pre body za lineup väčšiu váhu. Každý headliner má váhu rovnajúcu sa počtu doplňujúcich účinkujúcich + 1 za seba samotného. To by malo odzrkadľovať skutočnosť, že headliner je aspoň tak dôležitý, ako všetci jeho doplňujúci dídžeji. Očakáva sa, že ak užívateľ zadal daného dídžeja ako headlinera, je naozaj kvalitnejší než ostatní doplňujúci dídžeji. Ak by však takýto skutočne kvalitný dídžej (napr. podľa nejakých oficiálnych rebríčkov, alebo proste podľa názorov komunity) bol v systéme vytvorený relatívne neskoro, kedy by teoreticky slabší dídžeji už stihli získať vyššie levely a užívateľ by takéhoto neskoro vytvoreného dídžeja zvolil ako headlinera, znižoval by celkové bodové ohodnotenie lineupu kvôli svojej vyššej váhe, čo nie je vôbec žiaduce správanie. Preto, ak má zvolený headliner menší level, ako najlepší doplňujúci účinkujúci, jeho váha sa nastaví na 1.

Posledná dôležitá informácia pre tento výpočet je momentálny najvyšší level medzi interpretmi v celom systéme. Pomocou tohto čísla sa vypočítajú body jednotlivých účinkujúcich ako pomer levelu daného účinkujúceho k najvyššiemu levelu spomedzi interpretov v celom systéme. Výsledok za lineup sa nakoniec vypočíta ako vážený priemer bodov jednotlivých účinkujúcich. Názornejší je nasledujúci fiktívny príklad:

Predstavme si situáciu, že na hudobnej akcii vystupujú 4 dídžeji: Adam, ktorý má level 8, Boris, ktorý má level 5, Cyril má level 3 a Daniel má level 2. Adam je jediný headliner akcie. Jeho body budú mať váhu určenú ako súčet levelov ostatných dídžejov + 1 za seba samotného, teda 11. Povedzme, že najlepší interpret v systéme má level 12, takže Adam dostane 8 / 12, teda 0,67 bodu. Boris dostal 5 / 12, teda 0,42 bodu. Cyril 0,25 a Daniel 0,17 bodu. Tieto čísla dosadíme do vzorca (*viď. Vzorec 4.4*), kde *n* je počet headlinerov, V_h je váha headlinera, B_h sú body headlinera, *k* je počet doplňujúcich dídžejov a B_d sú body doplňujúceho dídžeja. Výsledný počet bodov je B * 10, teda pre náš príklad je to po zaokrúhlení 5,86 bodu.

$$B = \frac{\sum_{i=1}^{n}(V_h * B_h^i) + \sum_{i=1}^{k}(B_d^i)}{n * V_h + k} \tag{4.4}$$

4.4.3 Hudobné štýly

Ďalšie kritérium, ktoré treba zohľadniť pri automatickom filtrovaní je to, nakoľko sú pre daného užívateľa relevantné hudobné štýly danej akcie. Užívateľ získava tiež xp body, no tie sa mu narozdiel od interpretov a klubov rozdeľujú do hudobných štýlov a ukladajú do tabuľky *user_experience*. Podľa toho, aký má užívateľ levely za hudobné štýly, sa vypočítajú body každému štýlu, ktorý sa bude hrať na konkrétnej akcii. Tentokrát je podstatný najvyšší level medzi štýlmi daného užívateľa. Počet xp bodov *x*, ktoré užívateľ musí dosiahnuť pre level *n* je rovnaký, ako pri dídžejoch (*viď. Vzorec 4.3*). Dôležitá sú taktiež priority štýlov pri danej hudobnej akcii.

Každému štýlu, ktorý sa na akcii hrá, sa vypočíta bodové ohodnotenie ako pomer užívateľovho levelu za daný štýl k najvyššiemu levelu, aký tento užívateľ za nejaký štýl má. Ak ide o hlavný štýl akcie (teda jeho priorita je nastavená na 3), jeho váha je 4krát väčšia. Malo by to znamenať, že doplňujúce hudobné štýly sú na akcii len štvrtinovo podstatné. Ak ide o doplňujúci štýl, jeho váha ostane nastavená na 1. Výsledné body za relevantnosť hudobných štýlov sa opäť získajú ako vážený priemer ich jednotlivých bodov, ako ukazuje vzorec 4.5, kde *n* je počet hlavných štýlov akcie, B_h sú body hlavného štýlu, *k* je počet doplňujúcich štýlov a B_d sú body dopĺňajúceho štýlu. Výsledný počet bodov za štýly je B * 10.

$$B = \frac{\sum_{i=1}^{n}(4 * B_h^i) + \sum_{i=1}^{k}(B_d^i)}{n * 4 + k} \tag{4.5}$$

4.4.4 Kvalita klubu

Body za kvalitu klubu sa získavajú jednoducho ako pomer levelu daného klubu k najvyššiemu levelu medzi klubmi v systéme. Tento pomer je treba ešte vynásobiť 10. Počet xp bodov *x*, ktoré klub musí dosiahnuť pre level *n* udáva vzorec 4.6. Dôležitá je konštanta 5, ktorá je vyššia ako pri výpočte levelu interpretom alebo užívateľom, aby kluby nezískavali vyššie levely príliš rýchlo, pretože sa očakáva, že na akcie chodí veľa ľudí, ktorí ju následne hodnotia.

$$x = 5 * (3 * n + 2) * (n - 1) \tag{4.6}$$

Koľko xp bodov treba na dosiahnutie prvých 5 levelov pre kluby, užívateľov a interpretov pre ilustráciu ukazuje tabuľka 4.4.

4.4.5 Výsledné hodnotenie hudobnej akcie

Keď systém vypočítal body za všetky 4 kritéria, dosadí ich do výsledného výpočtu, ktorý je tiež riešený ako vážený priemer jednotlivých bodových hodnôt s príslušnými váhami, ktoré boli spomenuté na začiatku časti 4.4. Ak nemá klub zadané mesto, v ktorom sa nachádza, vzdialenosť sa do výsledného výpočtu nezahrnie.

Ako bolo načrtnuté v časti *4.3 Užívateľské rozhranie,* užívatelia majú možnosť si automatický filter prispôsobiť pomocou posuvníkov, z ktorých je možné každý nastaviť na 5 rôznych hodnôt, pomocou ktorých užívateľ systému hovorí, ako sú pre nich jednotlivé kritéria dôležité. Hodnoty by slovami boli reprezentované takto: *na kritériu vôbec nezáleží, málo záleží, stredne záleží, viac záleží* a *veľmi záleží*.

Ak kritérium pre užívateľa *nie je vôbec dôležité*, hudobné akcie dostanú zaň 10 bodov, teda akoby toto kritérium splňovali dokonale. Tým pádom sa mu zobrazia aj napríklad vzdialenejšie akcie. Navyše sa váha kritéria zmenší o polovicu a teda bude mať na výsledok menší vplyv.

V prípade, že na kritériu užívateľovi *málo záleží*, vynásobí sa počet bodov za toto kritérium konštantou 1,5. Maximálny počet bodov je však 10.

Keď je posuvník v strede, teda *na kritériu záleží stredne*, vypočítané body sa nezmenia. To je vlastne implicitná funkcionalita automatického filtru.

Ak *na kritériu viac záleží*, bude mať vo výslednom výpočte 1,5krát väčšiu váhu, no maximálne váhu 7, čo je najvyššia váha spomedzi váh jednotlivých kritérií. Ak teda akcia splňuje dané kritérium veľmi slabo, celkové hodnotenie bude ešte horšie a naopak.

To, že na danom kritériu *veľmi záleží*, znamená, že jeho váha bude rovnaká, ako tá najvyššia, teda váha vzdialenosti, ktorá je nastavená na 7.

Keď systém vypočíta všetky body a všetky váhy, dosadí ich do finálneho vzorca váženého priemeru a získa tak počet bodov pre danú akciu. Body sú spravidla v rozmedzí 0 až 10. Posledné pravidlo je jednoduché a vraví: Ak hudobná akcia dostala menej ako 5 bodov, odstráni sa z výberu. Užívateľovi sa teda zobrazia len tie akcie, ktorých hodnotenie presiahlo hranicu 5 bodov.

4.4.6 Spôsoby získavania xp bodov

Možností, ktorými sa dajú nadobudnúť xp body, či už ide o užívateľa, klub alebo interpreta, je niekoľko a u každého typu subjektu sa líšia.

Užívatelia majú xp body uložené pri každom hudobnom štýle zvlášť. Vždy keď užívateľ vykoná akciu, za ktorú získa xp body, musia sa mu v správnom pomere rozdeliť podľa toho, akú prioritu ktorý štýl pri danom uzli má. Uvediem príklad, z ktorého by to malo byť jasné. Povedzme, že chceme rozdeliť 16xp bodov medzi dva štýly s veľkou prioritou (určené číslom 3) a dva štýly s malou prioritou (1). Súčet priorít je 8. Rozdelíme teda 16xp bodov na 8 rovnakých častí, teda po 2xp bodoch. Keď má štýl prioritu 3, pridelia sa mu 3 takéto časti, v tomto prípade 6xp bodov, a keď má prioritu 1, tak len 1 časť, teda 2xp body. Jednoducho povedané, xp body sa medzi štýly rozdelia tak, že tie s veľkou prioritou dostanú trikrát viac xp bodov, ako tie s nízkou prioritou. Algoritmus je takto navrhnutý preto, aby bolo možné v budúcnosti pridať viacero priorít. Akcie, za ktoré užívateľ dostáva xp body sú vymenované v tabuľke 4.1.

Akcia vykonaná užívateľom	Počet získaných xp bodov
Výber, že sa zúčastní budúcej akcie	5
Výber, že sa možno zúčastní budúcej akcie	2
Kladné ohodnotenie akcie	10
Priemerné ohodnotenie akcie	4
Udelenie ocenenia za najlepšieho účinkujúceho	3
Pridanie hudobného setu	15
Pridanie hudobnej akcie	20

Tabuľka 4.1 – Možnosti užívateľa pre získanie xp bodov

V tabuľke sú skutočne len tie akcie, za ktoré užívateľ dostane xp body. Tieto xp body znamenajú to, nakoľko sa užívateľ v danom štýle „pohybuje“ resp. ako ho má rád, nakoľko sa mu venuje. Keď užívateľ vyberie napr. možnosť, že sa budúcej akcie nezúčastní alebo že sa nezúčastnil minulej akcie, nepridá mu to žiadne xp body, no aj napriek tomu sa mu ukáže „bublinka“, ktorá mu oznámi, že xp body získal. Je to malé klamstvo, ale motivuje to užívateľa k tomu, aby naopak on neklamal o svojej návštevnosti a to vďaka tomu, že bude mať pocit, že si akýmsi spôsobom pomohol, aj keď na akcii nebol. Keďže sa mu žiadne xp body nepridali, bolo by lepšie povedať, že si v skutočnosti len „neuškodil“, pretože by filter inak dostal o niečo mylnejšie informácie. Ak si užívateľ rozmyslí návštevu alebo hodnotenie, xp body sa správne zmenia aj jemu, aj prípadnému klubu, či interpretovi.

Interpreti dostávajú xp body za iných podmienok a nedelia sa im medzi štýly. Počet xp bodov je väčšinou na niečom závislý. Všetky tieto možnosti sú v tabuľke 4.2. Keď dídžeji hrajú v istý čas na istej akcii, sú vždy v tej dobe „nejako dobrí“, teda majú nejaký konkrétny level. Keď im užívateľ udelí cenu za najlepšieho dídžeja na konkrétnej akcii, znamená to v podstate, že dídžej bol lepší než všetci ostatní v lineupe. Získa za to xp body, ktoré sa z toho lineupu vypočítajú ako súčet levelov všetkých účinkujúcich (vydelený dvoma). Ako dídžeji získavajú časom stále viac xp bodov, aj levely im postupne zvyšujú. Mohla by teda nastať situácia, že by sa užívateľ „preklikal“ k nejakej starej akcii, ktorá sa konala niekedy dávno, odkedy sa levely daných účinkujúcich už zvýšili. Keby udelil jednému z účinkujúcich cenu len teraz, dostal by daný účinkujúci viac xp bodov. To by vlastne vravelo, že svojím výkonom predbehol lepších interpretov, než v tej dobe v skutočnosti boli. Aby sa tejto situácii zabránilo, súčet xp bodov všetkých účinkujúcich sa ukladá do databázy k danej akcii už pri jej vytvorení. Úplne ideálne by však bolo toto číslo aktualizovať napríklad v deň konania danej akcie.

Situácia	Počet získaných xp bodov
Interpret hrá na akcii v danom klube	Level klubu * 3
Interpret hrá na akcii v lineupe s viacerými interpretmi	Súčet levelov všetkých účinkujúcich z lineupu°
Dostane ocenenie za najlepšieho dídžeja akcie	Súčet levelov všetkých účinkujúcich / 2
Užívateľ ohodnotí kladne akciu, na ktorej interpret hral	2
Užívateľ ohodnotí priemerne akciu, na ktorej interpret hral	1

Tabuľka 4.2 – Situácie, za ktoré interpret dostáva xp body

° V situácii, keď interpret hrá na akcii, na ktorej je headlinerom, sa počet xp bodov, ktoré za to získa vynásobí koeficientom 1,5. V tejto situácii získa dídžej viacej xp preto, lebo byť zvolený za headlinera akcie znamená, že je dídžej naozaj kvalitný (alebo aspoň najkvalitnejší z daného linupu).

Kluby dostávajú xp body za najmenej situácií a tiež sa im nerozdeľujú medzi štýly. Sú vymenované v tabuľke 4.3.

Situácia	Počet získaných xp bodov
V klube sa koná hudobná akcia	Súčet levelov všetkých účinkujúcich na akcii
Užívateľ ohodnotí kladne akciu, ktorá sa konala v danom klube	5
Užívateľ ohodnotí priemerne akciu, ktorá sa konala klube	1

Tabuľka 4.3 – Situácie, za ktoré klub dostáva xp body

Level	Užívateľ, interpret	Klub
1	0	0
2	24	40
3	66	110
4	126	210
5	204	340

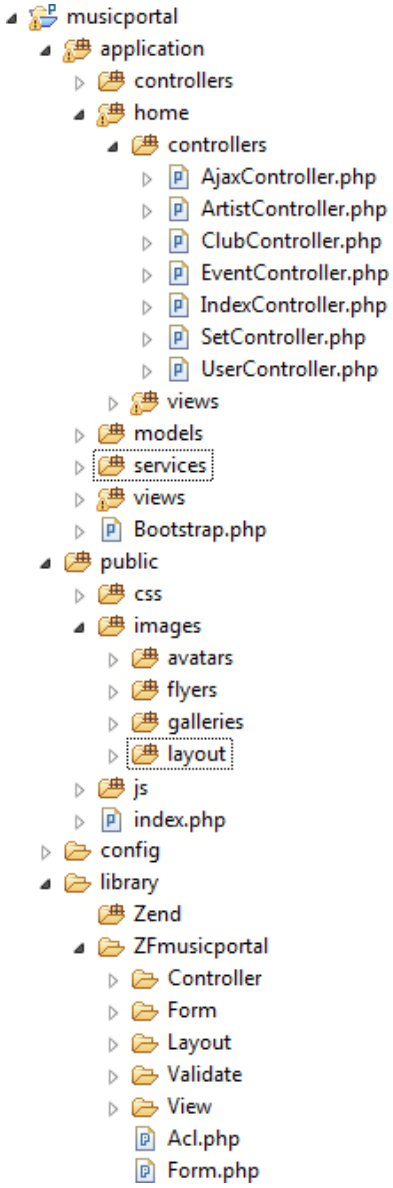
Tabuľka 4.4 – Počty xp bodov, ktoré musia subjekty mať na dosiahnutie prvých 10 levelov

5 Implementácia

Pre implementáciu som použil spomínaný PHP framework Zend. Ten odporúča a navrhuje, akým spôsobom pri vývoji aplikácie postupovať.

5.1 Adresárová štruktúra

Zend Framework napríklad odporúča aj niekoľko spôsobov, akú vytvoriť adresárovú štruktúru aplikácie. Tá je dôležitá pre ľahkú orientáciu v aplikácii, čím urýchľuje prácu. Zvolil som si jednu z odporúčaných štruktúr, ktorá mi pripadala najpoužiteľnejšia pre tento projekt (*viď. Obrázok 5.1 – Adresárová štruktúra*).



Obrázok 5.1 – Adresárová štruktúra aplikácie

5.2 MVC

Zend funguje na princípe MVC (Model, View, Controller) a odporúča dodržovať isté zásady. Riadil som sa podľa nich, pretože to pomáha udržovať kód aplikácie čitateľný a ľahko rozšíriteľný. Jedna z týchto zásad je udržovať *kontroléry* čo najkratšie a to tak, že všetka funkcionalita, ktorú využívajú sa umiestni do modelov. Týmto spôsobom ju môžem znovu využiť na iných miestach aplikácie.

View obsahuje prezentáciu stránok. V tomto komponente sú len značky definujúce vzhľad a sémantiku dokumentu a len sa vypisujú „eskejpované“ hodnoty premenných, definovaných v kotroléri. Nič sa tu nepočíta, všetky dáta by mali byť už pripravené.

Modely sa starajú o dáta a o ich interakciu s aplikáciou a databázou. Ja som si v modeloch nechal priestor len pre vzťahy, ktorými sú poprepájané tabuľky v databáze. Všetku manipuláciu s dátami som umiestnil do tzv. *servisných tried*, ktoré som si na tento účel vytvoril a umiestnil do adresára *services*. Pre každú databázovú tabuľku je práve jedna servisná trieda. Všetky servisné triedy dedia funkcionalitu z triedy *Service_Abstract*, ktorá má implementované základné prostriedky pre prácu s databázovými objektami pomocou funckií:

- `getObjectById()`
- `getObjectByNodeId()`
- `newObject()`
- `saveObject()`
- `deleteObject()`
- `getAllObjects()`
- `getNameIdPairs()`

5.3 Servisné triedy

Servisné triedy obsahujú v podstate všetku funkcionalitu aplikácie, na ktorú sa kontrolery len odkazujú volaním príslušných funkcií. Sú tu uložené konštanty a premenné, ktoré sa v aplikácií používajú ako napríklad váhy pre filter, identifikátory pre typy návštev a hodnotení, počty xp bodov za konkrétne situácie atp. Keby sa teda po dlhšom používaní aplikácie napríklad zistilo, že niektoré hodnoty xp bodov nie sú ideálne vyvážené, nie je problém ich na jedinom mieste zmeniť. Okrem premenných majú servisné triedy metódy pre rôznu manipuláciu s dátami, podľa toho, ako sa to pre danú situáciu hodí. Sú v nich aj rôzne pomocné funkcie napríklad na výpočet vzdialenosti medzi mestami alebo na vytvorenie odkazu pre vyhľadanie spojenia pomocou služby *idos.cz*.

5.3.1 Vyhľadávanie IDOS

Ak sa hudobná akcia koná v inom meste, než býva užívateľ, je očakávané, že ak sa túto akciu rozhodne navštíviť, bude si chcieť nájsť medzimestský spoj v deň konania akcie. Toto by mu mal uľahčiť odkaz pri každej takejto akcii odkazujúci na stránky idos.cz. Táto služba umožňuje na stránku umiestniť taký odkaz, ktorý má v sebe parametre, pomocou ktorých sa stránka idos.cz zobrazí s už vyplneným formulárom. Odkazu nastavujem parametre:

- `f` - východzie mesto, teda mesto, kde býva užívateľ
- `t` – cieľové mesto, teda mesto, kde sa nachádza klub
- `date` – dátum, kedy sa majú vyhľadať spoje, teda dátum konania akcie
- `time` – čas odkedy má v daný dátum vyhľadať spoje
- `lng` – jazyk stránok idos.cz

Ak je mesto v inom štáte než Česká republika, musí sa zaň v hranatých zátvorkách vložiť hviezdička a skratka štátu (napr. „[*A]" pre Rakúsko). Nakoľko služba nepoužíva dvojmiestne skratky štátov podľa štandardu, musí sa pri každom štáte do databázy ručne pridať táto skratka. značku pridávam aj pri českých obciach.

Parameter `time` nastavujem podľa toho, ako veľmi sú mestá od seba vzdialené. Od času, kedy akcia oficiálne začína odčítam 1 hodinu a ďalšiu jednu hodinu za každých 50km vzdialenosti. Ak nie je z čoho počítať vzdialenosť (jedine v prípade keď klub nemá určené mesto), čas sa nastaví o 5 hodín menší, než je čas začatia akcie. Užívateľ aj tak vie sám najlepšie, v aký čas chce na akciu ísť a zmení si ho podľa vlastného uváženia.

Parameter `lng` sa nastaví na `C` teda češtinu, ak je užívateľ buď z Českej alebo Slovenskej republiky. Inak sa nastavý na `E` teda angličtinu. Odkaz môže teda vyzerať napríklad nasledovne:
http://www.idos.cz/spojeni/?f=Wien%20[*A]&t=Brno%20[*CZ]&date=22.5.2010&time=15:00&lng=E

6 Možné vylepšenia

Tento projekt má ambície byť komplexným hudobným portálom s veľkým počtom návštevníkov na čo najväčšom území a čo najväčším počtom relevantných informácii. Malo by k tomu pomôcť viacero sekcií, ktoré by sa mohli v budúcnosti doimplementovať. Preto bol kladený veľký dôraz na spomínanú čitateľnosť a rozšíriteľnosť kódu. Bolo by nutné z geonames importnúť mestá a obce a ich zemepisné šírky a dĺžky ostatných krajín a užívateľské rozhranie preložiť do príslušných cudzích jazykov.

6.1 Fórum

Asi najzákladnejším rozšírením by bolo ponúknuť užívateľom možnosť vyjadriť sa. V prvom rade pomocou komentárov, ktoré by bolo možné pridávať k akciám, klubom, interpretom, setom ale aj samotným užívateľom. Ďalšie miesto pre vyjadrenie by užívatelia mali vo fórach, ktoré by si mohli sami vytvárať na rôzne témy, ktoré by boli rozdelené do kategórií.

6.2 Produkčná sekcia

Bolo by to miesto pre amatérskych producentov, ktorí by sem mohli uploadovať svoje hudobné výtvory. Tie by si mohli navzájom komentovať a hodnotiť, aby si tak navzájom vymieňali názory, postrehy a rady a tak sa mohli rýchlejšie zdokonalovať. Aby sa z užívateľa stal producent, ktorý by mohol uploadovať svoje pesničky, musel by najprv poskytnúť odkazy aspoň na tri svoje pesničky, ktoré by správcovia skontrolovali a podľa toho mu povýšili status na *producenta.*

6.3 Priatelia

Známa funkcionalita z množstva portálov. Využívali by sa napríklad na to, aby automatický filter pre užívateľa lepšie vyberal hudobné akcie. Čím viac priateľov by malo v pláne sa danej akcie zúčastniť, tým by bola akcia pre užívateľa zaujímavejšia.

6.4 Komunikácia užívateľov

Užívatelia by mohli spolu komunikovať napríklad pomocou internej pošty, prípadne nejakého jednoduchého, rýchleho chatu.

6.5 Lepšia administrácia

Užívatelia by mohli mať viacero typov práv. Z predchádzajúcich vylepšení vyplývajú role ako vlastník fóra, ktorý by mohol spravovať fórum, tj. mazať príspevky, menovať ďalších vlastníkov, udeľovať užívateľom *read-only (zákaz prispievať)*, alebo zákaz vstupu do svojho fóra. Rola producenta by zahrňovala privilégium pre pridávanie pesničiek a hodnotenie pesničiek ostatných producentov.

Ak by užívateľ pridal nevhodný obsah, ostatní užívatelia by mohli na takýto príspevok upozorniť správcov, ktorí by tento príspevok upravili alebo prípadne zmazali. Užívateľovi by za to hrozilo odobratie práv pre pridávanie daného typu príspevku, alebo by mu bol úplne zakázaný prístup na stránku.

6.6 Fotogalérie

Pre užívateľov hudobného portálu zameraného na tanečné akcie sú veľmi dôležité fotoreportáže z daných akcii. Preto by nemohla táto funkcionalita chýbať. Galérie by sa sťahovali ku konkrétnym akciám a boli by (rovnako ako samotné fotky) riešené ako uzly (`nodes`) a teda by sa dali komentovať, prípadne hodnotiť.

6.7 Kolekcia užívateľa

Išlo by o menšie „okno“ , ktoré by užívateľ mal stále dostupné, kde by mohol *drag and dropom* prenášať všetky typy uzlov. Podľa typu uzla by bola vykonaná príslušná akcia. Napríklad ak by šlo o budúcu akciu, užívateľ by tým automaticky systému povedal, že sa ju chystá navštíviť. Ak by to bol komentár, alebo fotka, znamenalo by to, že sa mu páči a udelil by tým uzlu svoje hodnotenie. Ak by šlo o užívateľa, požiadal by ho tým o priateľstvo. Všetky takto prenesené uzly by mal zároveň užívateľ na jednom mieste, prehľadne roztriedené, takže by nemusel tak často používať globálne vyhľadávanie (čo by bola ďalšia funkcionalita).

Teoreticky by bolo možné pridať ešte o dosť viac funkcií, no to by záležalo na tom, či by sa portál ujal a užívateľom dostatočne pozdával, aby sa naň vracali. Mohlo by ísť napríklad o pridávanie komunitných noviniek, blogov atp.

7 Záver

Podarilo sa mi navrhnúť a vytvoriť základ pre webový portál špecializujúci sa na elektronickú tanečnú hudbu. Jeho hlavná funkcionalita spočíva v zbieraní informácií od užívateľov, na základe ktorých im je implementovaný systém schopný odporučiť také hudobné akcie, ktoré sa svojím zameraním a inými vlastnosťami najviac približujú predstavám užívateľa. Pre návrh hodnotiaceho systému a portálu všeobecne som využil skúsenosti z oblasti elektronickej hudby, a preto verím, že som ostatným ľudom s podobnými záľubami sprostredkoval rozumnú a príjemnú webovú aplikáciu. Skutočnú presnosť automatického filtru však ukáže až reálne nasadenie portálu do praxe, kedy ho vyskúšajú počas dlhšieho obdobia viacerí užívatelia. Tých by mohol na portál prilákať príjemný vzhľad a ovládanie prípadne niektoré možné vylepšenia, kvôli ktorým som sa snažil kód spraviť čo najľahšie rozšíriteľný.

Túto bakalársku tému som si vybral preto, aby sa niekedy mohla reálne presadiť a používať a prispieť tak hudobnej komunite. Chcel som ale taktiež nadobudnúť novú prax v oblasti vytvárania informačných systémov, ich optimalizácii a zdokonaliť sa v technológiách ako AJAX, Zend Framework a JavaScript, čo sa mi rozhodne podarilo. Táto práca ma obohatila aj o skúsenosti s *navrhovaním* informačných systémov, nie len vytváraním aplikácie a implementovaním funkcionality podľa už hotových špecifikácií. Osvojil som si tiež niekoľko metód, ktoré pomáhajú vyvíjať software efektívnejšie, ako napríklad odstraňovanie chýb hneď ako sa naskytnú alebo lepšie odhadovať čas, ktorý treba na vytvorenie určitého celku, čo sa mi všetko do budúcna bude hodiť

Literatúra

[1] *HTML* [online]. Last modified on 8 May 2010. [cit. 2010-05-09]. Dostupné na URL: <http://en.wikipedia.org/wiki/HTML>.

[2] *XHTML* [online]. Last modified on 9 May 2010. [cit. 2010-05-09]. Dostupné na URL: <http://en.wikipedia.org/wiki/XHTML>.

[3] Stansberry, G. *30 CSS Best Practices for Beginners* [online]. Last modified on 16 September 2010. [cit. 2010-05-09]. Dostupné na URL: <http://net.tutsplus.com/tutorials/html-css-techniques/30-css-best-practices-for-beginners/>.

[4] *Cascading Style Sheets* [online]. Last modified on 3 May 2010. [cit. 2010-05-09]. Dostupné na URL: <http://en.wikipedia.org/wiki/Cascading_Style_Sheets>.

[5] *Blueprint Framework* [online]. Dostupné na URL: < http://www.blueprintcss.org/>.

[6] *Browser Sandbox* [online]. Dostupné na URL: <http://spoon.net/browsers/>.

[7] Lednár, M. *Príručka programátora - Prehľadný sprievodca jazykom JavaScript 1.5+*. MLD Group, Bratislava, 2009. ISBN 978-80-89448-02-9.

[8] *JavaScript* [online]. Last modified on 7 May 2010. [cit. 2010-05-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/JavaScript>.

[9] Chaffer, J., Swedberg, K. *Learning jQuery 1.3.* 2. vyd. Packt Publishing, 2009-1-13. 336 s. ISBN 978-1847196705.

[10] Achour, M., Betz, F., Dovgal, A., et al. *PHP Manual* [online]. Last modified on 10 May 2010. [cit. 2010-05-10]. Dostupné na URL: <http://www.php.net/manual>.

[11] *PHP* [online]. Last modified on 10 May 2010. [cit. 2010-05-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/PHP>.

[12] *Zend Framework Manual* [online]. 2010. [cit. 2010-05-11]. Dostupné na URL: <http://framework.zend.com/manual>.

[13] Reyey, J. *Discussing PHP Frameworks: What, When, Why and Which?* [online]. Last modified on 26 July 2009. [cit. 2010-05-10]. Dostupné na URL: <http://www.noupe.com/php/discussing-php-frameworks.html>.

[14] Helman, D. *Model-View-Controller* [online]. Last modified on 14 May 1998. [cit. 2010-05-10]. Dostupné na URL: <http://ootips.org/mvc-pattern.html>.

[15] Crane, D., Pascarello, E., James, D. *AJAX In Action*. Mannig Publications Co., Greenwich, 2006. ISBN 1-932394-61-3.

[16] Crockford, D. *Introducing JSON* [online]. Last modified on 28 May 2009. [cit. 2010-05-10]. Dostupné na URL: < http://www.json.org/>.

[17] *Secure Hash Algorithm* [online]. Last modified on 7 May 2010. [cit. 2010-05-11]. Dostupné na URL: <http://en.wikipedia.org/wiki/Secure_Hash_Algorithm>.

[18] Shreve, J. *5 Helpful Tips for Creating Secure PHP Applications* [online]. Last modified on 26 December 2008. [cit. 2010-05-11]. Dostupné na URL: <http http://net.tutsplus.com/tutorials/php/5-helpful-tips-for-creating-secure-php-applications/>.

[19] Evans, C. *PHP Security Tips* [online]. 2007. [cit. 2010-05-11]. Dostupné na URL: <http://devzone.zend.com/tag/Security_Tips>.

[20] *The Cross-Site Scripting (XSS) FAQ* [online]. 2002. [cit. 2010-05-11]. Dostupné na URL: <http://www.cgisecurity.com/xss-faq.html>.

[21] *Yahoo! YSlow* [online]. [cit. 2010-05-11]. Dostupné na URL: <http://developer.yahoo.com/yslow/>.

[22] *Page Speed* [online]. [cit. 2010-05-11]. Dostupné na URL: <http://code.google.com/speed/page-speed/>.
[23] *Firebug* [online]. Dostupné na URL: <http://getfirebug.com/>.
[24] Theurer, T. *Performance Research, Part 2: Browser Cache Usage – Exposed!* [online]. Last modified on 4 January 2007. [cit. 2010-05-11]. Dostupné na URL: <http://yuiblog.com/blog/2007/01/04/performance-research-part-2/>.
[25] Way, J. *30 HTML Best Practices for Beginners* [online]. Last modified on 14 May 2009. [cit. 2010-05-11]. Dostupné na URL: <http://net.tutsplus.com/tutorials/html-css-techniques/30-html-best-practices-for-beginners/>.
[26] Azad, K. *How To Optimize Your Site With GZIP Compression* [online]. Last modified on 4 April 2007. [cit. 2010-05-11]. Dostupné na URL: <http://betterexplained.com/articles/how-to-optimize-your-site-with-gzip-compression/>.
[27] *Google Code* [online]. 2010. Dostupné na URL: <http://code.google.com/apis/ajaxlibs/>.
[28] Smith, J. H. *Improve your jQuery - 25 excellent tips* [online]. Last modified on 14 December 2008. [cit. 2010-05-12]. Dostupné na URL: <http://www.tvidesign.co.uk/blog/improve-your-jquery-25-excellent-tips.aspx>.
[29] *Google's SEO Starter Guide* [online]. Last modified on 12 November 2008. [cit. 2010-05-12]. Dostupné na URL: <http://googlewebmastercentral.blogspot.com/2008/11/googles-seo-starter-guide.html>.
[30] *Diagram prípadov použitia* [online]. Čas poslednej úpravy 11. November 2009. [cit. 2010-05-15]. Dostupné na URL: <http://sk.wikipedia.org/wiki/Diagram_pr%C3%ADpadov_pou%C5%BEitia>.
[31] *English country names and code elements* [online]. 2010. [cit. 2010-05-16]. Dostupné na URL: <http://www.iso.org/iso/english_country_names_and_code_elements>.
[32] *IDOS - Vyhledání spojení* [online]. Dostupné na URL: <http://jizdnirady.idnes.cz/vlakyautobusy/spojeni/>.
[33] *Great-circle distance* [online]. Last modified on 1 May 2010. [cit. 2010-05-16]. Dostupné na URL: <http://en.wikipedia.org/wiki/Great-circle_distance>.
[34] *GeoNames* [online]. [cit. 2010-05-16]. Dostupné na URL: <http://www.geonames.org/>.

Zoznam príloh

Príloha 1. CD so zdrojovými kódmi aplikácie a elektronickou verziou tejto technickej správy